

IMPORTANT NOTICE - PREMIER PUBLICATIONS products and printed matter appertaining to the use thereof are copyright of PREMIER PUBLICATIONS. Copying of the whole or in part in any medium, other than for the personal use of the original retail purchaser is strictly prohibited.

We are obliged to pursue an active policy against infringements of our copyrights. PREMIER PUBLICATIONS offers a substantial reward for information concerning copyright infringements.

#### GUARANTEE AND PRODUCT LIABILITY

Quality of materials and workmanship guaranteed. Claims under guarantee must be accompanied by the program cassette/disk/EPROM. Whilst every care has been taken in the preparation of this product, PREMIER PUBLICATIONS will not be liable for claims of loss or of loss of profit, howsoever caused. It is the responsibility of the purchaser to ensure suitability for purpose. This notice does NOT affect the statutory rights of the original retail purchaser.

BASIC 5 was commissioned by PREMIER PUBLICATIONS and written by P RIHAN.

COPYRIGHT 1981 P RIHAN &

PREMIER PUBLICATIONS.

## B A S I C 5

### INSTRUCTION MANUAL

PREMIER PUBLICATIONS

LEADERS IN UK101/OHIO SOFTWARE

## INSTALLATION

### EPROM VERSION

The EPROM locates at 9000hex. To initialise BASIC 5, turn on the computer, go through the cold start routine, enter the monitor and type:-

9200G

and the following message will appear

\* BASIC 5 \* V1.x

(C) 1981 P.RIHAN/PREMIER

Ready

The modified prompt is a reminder that BASIC5 is in residence. RESET Warm start will retain BASIC 5, a cold start will de-activate it.

If you are using PREMIER's BASIC 4 EPROM, BASIC 5 will be automatically initialised during the coldstart routine.

To engage BASIC 5 from BASIC without entering the monitor

POKE11,0:POKE12,146:USR(X)

If this method is used from program, control returns to the command mode and program execution ceases.

If you are also using PREMIER's TOOLKIT 1 or 2, initialise BASIC 5 first, then use the &GD\$ command to call in TOOLKIT via the appropriate entry point (8050 for TK1 and 8000 for TK2).

### DISK VERSION

To run the disk version, load your normal operating system, then put in the disk containing BASIC 5 and RUN"BASIC5".

Once booted, BASIC 5 is entered by RESET warm start. This will give the same initialisation message as the EPROM version.

## BASIC 5

BASIC 5 is a powerful and flexible extension to OSI BASIC as used in the various OHIO and UK101 machines. Many extra BASIC keywords are provided covering graphics, print formatting, CEGMON windows and general programming commands including input and output.

All commands are accessible from program and are used in lines as if they were inbuilt BASIC 'keywords'. They are all available in the immediate mode except &PRNTUSNG and &INAT which use the input buffer to format their text. In most circumstances (see later), BASIC 5 commands must be the first item in any statement (1st in a line or after '&') and may not be used within an expression or an IF...THEN statement (though they may form one of a series of statements after 'THEN'.) The 'OK' prompt is replaced in BASIC 5 by the more accurate 'Ready'!

The commands available are shown below - they are followed by a detailed explanation of each command.

### GRAPHICS

&SCR

&SET

&TEST

&VLIN

&HLIN

&BLK

### GENERAL

&GD or &GD\$

&GET

&GS

&GT

&RD

&INAT

&PUTAT

### FORMATTING

&PRNTUSNG

&"

&NI

&NI\$

&CNI

&CNI\$

The '&' shown in front of each command is essential. It flags to BASIC 5 that there is a command to decode. Without it, a syntax error will be generated.

In the notes below, the following notation is used to signify syntax.

< > a valid BASIC expression. Eg 10, A, N\*5, "WORD", B\$, etc

VAR means a valid variable name - use with care

\$HHHH is a four digit hex address.

## GRAPHIC COMMANDS

All graphic commands (plus INAT and PUTAT) use a row/column system whereby 0,0 is the bottom left corner of the screen. Eg:- to place the start of the graphics at row 7, column 12 you should input 7,12 at the appropriate place in the command word.

### SCR

&SCR<CHR#> will fill the screen with the character specified. The character must have a value 0-255 and can be either a number or a variable. The screen is filled in around 0.025sec. Try this example.

```
10 FOR N=0 TO 255
20 &SCRN
30 NEXT
```

You'll find that this passes very quickly, so add

```
15 FOR D=1 TO 400: NEXT
```

to slow the process down a little! &SCR32 in the immediate mode will clear the screen - useful if you do not have CECMON.

### SET

&SET <ROW>, <COLUMN>, <CHR#> will set the CHR# specified at row, column

You are limited to the screen area (or current window). Try:-

```
10 &SCR32: FOR N=0 TO 20
20 &SET8,N,161
30 NEXT
```

To SET an individual point on the screen at row 7, column 5, with a letter A:-

```
10 &SET7,5,65 (REM 65 is ASCII for 'A')
```

### TEST

&TEST <ROW>, <COLUMN>, VAR returns the number of the character located at the row, column in the variable used. The variable must be numeric. Try:-

```
10 &SCR46
20 &TEST5,5,P
30 PRINT P
```

### VLIN

&VLIN <ROW>, <COLUMN>, <LENGTH>, <CHR#> will set a vertical line from row, column chosen of length specified using chosen character. The sum of row and length must fit within the current window.

```
10 &VLIN0,5,16,161
```

will draw a vertical line from the bottom of the screen to the top. Variables may be used to specify row, column, etc. For example:- row can be 12, N, N\*4-1, etc. A function call error will be generated for values out of range.

### HLIN

&HLIN is the same as &VLIN, except horizontal lines are generated.

```
10 &SCR32
20 &HLIN0,0,20,132
```

will draw a narrow line along part of the bottom of the screen.

### BLK

&BLK <ROW>, <COLUMN>, <LENGTH>, <HEIGHT>, <CHR#>. This will set a block from row, column (bottom left) of height specified using the chosen character. For example

```
10 &BLK0,0,20,10,161
```

will produce a large block on screen very quickly.

*Diagonal lines (on line zero)*

*Poke 316, 65 — right Diagonal*

*Poke 316, 63 — left diagonal*

*\* Always reset value back to 64 for normal BS use.*

## FORMATTING

A powerful 'PRINT USING' command is included in BASIC 5. It allows the user to form output formats for strings and variables. Two commands are used:- &" ##.### to form the 'image' for output, and &PRNTUSNG to access the format. The image for output may contain information along with the actual image. A 'print list' may follow &PRNTUSNG, as shown below. Each 'image' may contain any number of formats or may even be empty. Exponential numbers cannot be printed.

```
&" TEXT IMAGE TO BE PRINTED ##.###
```

defines the image to be printed. This comprises a text string containing a number (no limit) of 'formats' (####.### in the above example), into which variables or strings may be printed. The image may be of any length up to the current screen width (only checked when called) or empty. The image will be used again and again until the print list is exhausted and/or printing will stop at the first unfilled format. If you print an empty image or an image without formats (Eg a title), your print list must also be empty.

Formats possible are

```
#### for integer numbers (decimals are truncated) or strings left justified and
      truncated on the right.
####.## for decimal numbers justified on the decimal point. Trailing zeros are
      printed, strings as above BUT they count the point as a character slot.
>### integer numbers as before but leading zeros are printed, strings are right
      justified and left truncated. There must be at least one '#' before and after
      the decimal point.
>###.## as before but leading zeros are printed. Strings are treated as all character
      slots, but right justified.
```

The format may contain any number of '#' or groups of '#' up to the width limit of the screen.

The format is accessed by the command &PRNTUSNGx: where x is the line number containing the format (x may be a number, an expression or a variable). Note that the colon after the line number is essential as it defines the start of the 'print list'.

```
10 &"THE ANSWER IS ####.##
20 X=21456.9
30 &PRNTUSNG10:X,34.2,"WRONG"
```

Try the above example and examine the output generated. As you will see, the three lines of output are generated by the 'print list' on line 30 (after the colon).

It is also possible to modify the formats from program. If you define a string as "###.##" then print it into a format (it obeys the standard rules for strings in that format), then the string will form a format into which the next item in the print list will be printed. This does however mean that you cannot use " or # in strings to be printed this way, except deliberately!

When compiling your print list, do NOT use any terminator to the list such as a comma or semi-colon.

As can be seen from the above instructions, PRINT USING is a very powerful, but rather complex command. You are advised to look at the demonstration program supplied closely before attempting to use the function if this is your first encounter with it. It may also be helpful to consult any books on BASIC that you may have which mention PRINT USING.

ERROR MESSAGES resulting from mistakes in the use of PRNTUSNG require careful interpretation. If you attempt to call line 20 as an image and there is no image there, the normal SYNTAX error will be generated, giving the error as in line 20 (which, although not an image, may be an otherwise valid line of BASIC). You will therefore have to find the line which is calling line 20 as an image to correct the error.

## WI and CWI

These commands have been included so that users of CEGMON can easily manipulate their screen 'windows'.

&WI will reset BASIC 5's internal pointers to the current window, either on start up or after &CWI. (The routine is called during initialisation). Note that after using &WI, all error checking and function limits (especially of the graphic commands) will now refer to the current chosen window - be sure that you want this!

&CWI <TOP>, <BASE>, <SCREEN WIDTH>  
&CWI\$ <HHHH>, <HHHH>, <WIDTH> )

These commands allow you to alter the window to anywhere in memory and to any size up to 2k. The window is set from top to base with width as specified. Either or both addresses may be decimal or hex, but the width must be decimal. An expression may replace a decimal value.

The base must line up directly below the top value chosen and it must be below or equal to it or an FC error will be called. The cursor automatically homes in the new window. A knowledge of the screen addresses is needed to get the most out of this command. UK101 users try:-

```
10 &CWI53264,53264+(64*8),25
```

Now LIST your program !SUPERBOARD users with a 24x24 screen try

```
10 &CWI53285,53285+(29*32),23
```

and you should have more lines to look at!

Superboard and C1E users please note - in the 24x24 mode BASIC 5 sets its line increment (LL) to 32 (\$20). If you want to use 48 width or the Screen Enhancement Kit you must poke LL (at \$013C) with 64 (\$40) or restart from reset in the new mode. Otherwise &CWI and &WI will not work and screen based commands will produce some interesting garbage!

Cassette based users please note that there is no built-in protection to this command and indiscriminate use of &CWI could cause you to overwrite BASIC 5 if you aren't careful.....

## PUTAT

&PUTAT <ROW>, <COLUMN>, <Max LENGTH>, STRING

This useful function allows you to print a string anywhere on screen without causing scrolling or upsetting adjacent graphics. Row and column are used to decide the start position of the string with 0,0 being the bottom left of the screen. The 'length' parameter is needed to give a maximum field in which the string may be printed. This will generally be the space left on a line from the beginning of the string to the end of that line. Eg:- if the screen width is 48 and the string begins at column 23, the maximum 'field' left is 48-23=25, so 25 would be entered into the <Max length> box. If the field is too short for the string being printed, the string is truncated to the right. Try:-

```
10 A$="this is a string!"
20 &SCR46
30 &PUTAT5,2,22,A$
```

## INAT

&INAT <ROW>, <COLUMN>, <LENGTH>, VAR

This is a complementary function of PUTAT and allows you to input a string up to the defined length starting at row, column. In use, a cursor appears defining the next character space. Anything may be entered, including any graphics obtainable from the keyboard.

When the maximum input length is reached, the routine loops until RETURN or RUBOUT is pressed. RUBOUT has its normal function. Try:-

```
10 &INAT10,10,8,A$
20 &PUTAT8,10,15,A$
```

## GENERAL COMMANDS

&GO and &GOS

These two commands allow you to call a machine code routine from BASIC in either dec or hex notation. The address must be in the range 0 - 65535 or four hex digits. A syntax error will be generated for a decimal value out of range, BUT if a hex value is incorrectly input, (too few digits, for instance) you could end up anywhere ! Try:-

```
10 &GOSFD00:P=PEEK(531):?P::GOTO10
```

Exit the routine with CTRL C.

GET

&GET <VAR>

This is a much needed non-halting GET-KEY routine. It returns its value in VAR, which must be a string. Try:-

```
10 &GETK?:?K?:GOTO10
```

GT and GS

&GT<LINE> and &GS<LINE>

These two commands act exactly like GOTO and GOSUB, but <LINE> can be a variable or an expression, thus giving variable control of program sequence and allowing 'labels' to be used for subroutines. Try:-

```
10 X=999:&GSX:GOTO10
999 &SCR255*RND(8)+1:RETURN
```

RD

&RD<number>< VAR

This command will read through DATA lists to the <number> value chosen, with VAR being the recipient of the next piece of DATA. Note that the maximum value of <number> is 255. Try:-

```
10 DATA 1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j
20 RESTORE:&RD10,A:PRINT A
```

## GENERAL COMMENTS

All of these routines are fully error trapped (except where specified). Generally, if a parameter is out of range a FC error is called. If incorrect syntax or use is found SN error will occur, and if strings and numerics are mixed TM errors occur. The error to avoid at all costs is an OM error (referring to stack memory). Unfortunately, by the time this is called, the stack will have overwritten BASIC 5's 'permanent' store at \$0134 - \$013F. Three methods are available for recovery:-

- 1/ &GOS9200
- 2/ RESET W - but this will kill anything using I/O vectors
- 3/ RESET M &GOS9200 Use this if the zero-pointer has crashed but you may have to cold-start anyway.

This would only be necessary if an endless &GS or GOSUB loop had occurred or too many FOR...NEXT loops had been nested - a fairly rare occurrence!

## BASIC 5 and TOOLKIT

Users of PREMIER's TOOLKIT should note that the two products are completely compatible. However, RENUM will NOT renumber a BASIC 5 program containing &GTx, &GSx, and PRNTUSNCx. These will have to be renumbered manually. (Use the FIND routine to make this quick and simple).

Locations used by BASIC 5 include the input buffer, \$0011, \$0012, and \$00F0 - \$F3 in zero page for text formatting and temporary pointers. Permanent pointers (including LL - see text) are held in page 1 at \$0134 - \$3F.