YOUR SPECTRUM SOFTWARE

+ SPRITE DESIGNER

# MEGA BASIC

# CONTENTS

# GETTING STARTED

Loading *YS MegaBasic* is simplicity itself. Just type: **LOAD** “”
The program will then load and auto-run. *YS MegaBasic* also comes with another program called *Sprite Designer* that lets you design sprites easily for later use in your own programs. But make sure that you load *YS MegaBasic* before loading the Sprite *Designer,* if you want to transfer the *Sprite Designer* program to Microdrive, just select the ‘copy’ option from the main menu. *YS MegaBasic* also has a built-in transfer option for Microdrive owners — as soon as it's loaded, it’ll ask you if you want to transfer *YS MegaBasic* to Microdrive; press the ‘M’ key if you do — if not, press any other key.

Before making a start with *YS MegaBasic,* here are a few guidelines that’ll help you with your programming. First, because *YS MegaBasic* lives in RAM, you must be careful not to corrupt the code — POKEing around may cause crashes! Secondly, so that all of *YS MegaBasic*’s features could be packed into the limited program space, some error checking features have been left out. This means that you’ll have to take care when you’re typing in your programs. Make sure that command parameters are in the correct ranges and keywords are correctly spelt or they’ll be accepted as Procedures. You’ll find that checking your typing will save you a lot of time in debugging your programs anyway, and it’s a small price to pay for the extra K! Finally, remember that *YS MegaBasic* doesn’t use the normal Spectrum streams for printing, so don't use the following commands: ‘**CLEAR #**’, ‘**OPEN # 2**’, ‘**CLOSE # 2**’
or you’ll crash the system.
If you follow these few simple guidelines, you shouldn’t experience any problems with *YS MegaBasic.*

## THE KEYBOARD

Once *YS MegaBasic* has loaded, you’ll be greeted by a short copyright message and an inverse space in the bottom left-hand corner of the screen; this is the new cursor and it indicates where your input will be appearing. With *YS MegaBasic* the whole of the screen is now used for input instead (as with the normal Spectrum) of the screen being split into two parts.

Try typing a few characters on the keyboard and you’ll notice that the keys aren’t producing their usual keywords ... instead you see just single characters. From now on, you’ll have to type out each command in full rather than use the infamous keywords — a facility which transforms your Spectrum keyboard into something approaching that of a ‘normal’ computer

Although removing the keyword system has many advantages, the

change does have a drawback. Certain commands such as ‘PRINT’ could be typed in just by pressing ihe ‘P’ key, whereas now you'll have to type out ‘P’, ‘R’, ‘I’, ‘N’ and ‘T’; for that reason, *YS MegaBasic* allows you to abbreviate many of the keywords.

Here follows a complete list of keywords and their new abbreviations; you can assume that keywords omitted from the list cannot be abbreviated and therefore must be typed out in full. Also note that an abbreviated keyword must finish with a fullstop; for example, the abbreviation for ‘CONTINUE’ is ‘CON’.

| | | | |
|---|---|---|---|
| **A.TTR** | **ER.ASE** | **ME.RGE** | **RES.TORE** |
| **BE.EP** | **E.XP** | **M.OVE** | **RET.URN** |
| **B.IN** | **FL.ASH** | **NE.XT** | **R.ND** |
| **BO.RDER** | **F.ORMAT** | **N.OT** | **SA.VE** |
| **BR.IGHT** | **GOS.UB** | **OP.EN#** | **S.CREEN$** |
| **CH.R$** | **G.OTO** | **OV.ER** | **ST.R$** |
| **CI.RCLE** | **I.N KEY$** | **PA.PER** | **T.AB** |
| **CLE.AR** | **INP.UT** | **PAU.SE** | **TH.EN** |
| **CL.OSE#** | **INV.ERSE** | **PE.EK** | **U.SR** |
| **C.ODE** | **L.EN** | **PL.OT** | **V.AL$** |
| **CON.TINUE** | **LI.NE** | **P.OINT** | **VE.RIFY** |
| **DA.TA** | **LL.IST** | **PR.INT** | |
| **D.EF FN** | **LP.RINT** | **RA.NDOMIZE** | |
| **DR.AW** | **LO.AD** | **RE.AD** | |

Remember thai if you’re going to be typing commands such as ‘GO TO’ and ‘OPEN #’ in full, you'll have to remember to insert the necessary spaces; the computer will understand ‘GO TO’ but ‘GOTO’ will result in a “Syntax error” message being flashed on-screen.

The bottom line of the display is used to indicate the current mode of the cursor. Below is a table showing the normal Spectrum modes, along with the relevant on-screen message you’d expect on the bottom line.

| Cursor mode | Bottom line display |
|---|---|
| **‘L’** | CAPS OFF |
| **‘C’** | CAPS ON |
| **‘E’** | CAPS OFF EXTENDED |
| **‘G’** | CAPS OFF GRAPHICS |

The cursor’s mode is changed in the usual way — using the Caps Lock, Graphics. Caps Shift and Symbol Shift keys.

## THE EDITOR

The line editing capabilities have been greatly enhanced by *YS MegaBasic* and, as a result, some keys produce different effects to those found on the standard Spectrum.

| | |
|---|---|
| **EDIT** | Copies the current program line into the input line, ready for editing. |
| **TRUE VIDEO** | Deletes the whole of the input line. |
| **INVERSE VIDEO** | Deletes the character to the right ol the cursor. |
| **CURSOR LEFT** | Moves the cursor left one character. |
| **CURSOR DOWN** | Moves the cursor down one line within the input line. |
| **CURSOR UP** | Moves the cursor up one line within the input line. |
| **CURSOR RIGHT** | Moves the cursor right one character. |
| **DELETE** | Deletes the character to the left of the cursor |
| '<=' | Moves the cursor to the beginning of the input line. |
| '<>' | Deletes all the characters from the cursor to the end of the line. |
| '>=' | Moves the cursor to the end of the input line. |
| **SCREEN$** | Displays the automatic listings (output to window one). The top line in the listing is the current line. |
| **OR** | Moves the current line up one and displays the automatic listing. |
| **AND** | Moves the current line down one and displays the automatic listing. |
| **STOP** | Moves the 'copy' cursor left one character square. |
| **NOT** | Moves the 'copy' cursor up one line. |
| **STEP** | Moves the 'copy' cursor down one line. |
| **TO** | Moves the 'copy' cursor right one character square. |
| **AT** | Copies the character at the 'copy' cursor to the 'input' cursor. This command can also be used lo copy characters which are at standard size; it will not copy 64 column characters, double height characters or double height/double width characters. |
| **OVER** | Moves the 'copy' cursor to the next window. |
| **INVERSE** | Resets the 'copy' cursor to the top left of the current window. |

Note first, however, that the screen is split up into three different sections or windows — for more information on windows see page 6 and each is used for a specific task: window zero displays user input and error messages; window one displays automatic listings produced by the line editor; window two displays all output produced by Basic commands; and window three is used by the front-panel, see page 20. When *YS MegaBasic* is first loaded you may think there's only one window on-screen; in fact, there are four, all of which overlap.

Note too that a second cursor may be used to copy text from another part of the screen to the 'input' cursor. This cursor appears as a flashing square on-screen and can be moved around the current window via a number of pre-programmed control keys. The 'copy' cursor will only function in windows zero, one and two.

As well as using Caps Shift '1' to edit the current line it's possible to edit any line in the program using the 'EDIT_' command. This command is followed by a numeric expression that shows which line is to be edited. If the required line doesn't exist, then the next program line is used; if there's no next line then the "Line not found" error message is displayed on-screen.

## USER-DEFINED KEYS

So that you don't have to keep on typing in repetitive commands, it's possible to program the top row of keys to produce strings of up to 255 characters. To program a key, you use the 'KEY_' command which requires two parameters. The first defines the number (0-9) of the key to program, and the second provides the string to 'put in' that key. As usual with *YS MegaBasic* the parameters can be either immediate values or variables. Placing an 'ENTER' character at the end of the string (usually using '+CHR$(13)') enables the key to execute itself automatically when activated. You can activate a key by going into 'EXTEND' mode and pressing the key while holding down 'SYMBOL SHIFT'. To save you defining 'RUN' and 'LOAD' keys, *YS MegaBasic* already includes them:

| | |
|---|---|
| **VERIFY** | Comprising the string 'RUN+CHR$ 13'. this key runs the current program in memory. |
| **VALS** | Comprising the string 'LOAD "":RUN+CHR$ 13', this key loads and then runs lhe next program on tape. |

When you've defined your keys you can save them to tape using SAVE "filename" CODE 59956,2560 (or SAVE *"m";1 ;"filename" CODE 59956,2560 for Microdrive). This lets you restore them in one go next time you load up *YS MegaBasic* using LOAD "filename" CODE (or LOAD *"m";1;"filename" CODE).

### CONTROL KEYS

During RUN-time the Space key is used as a new Shift key. Consequently, to obtain a space between characters, both the Symbol Shift and Space keys must be pressed simultaneously. The Space key — when used together with another key — will from now on be referred to as the Control key. Here follows a number of Control functions available during RUN-time.

| | |
|---|---|
| **CONTROL F** | Calls up the front-panel (see page xx for more details). |
| **CONTROL E** | Stops program execution, prints the "Escape" message and returns to the line editor. |
| **CONTROL R** | Resets the YS MegaBasic system. It stops program execution, sets BORDER, PAPER and INK to their start-up values, prints a copyright message and returns to the line editor. This function does not erase any Basic programs in memory. |

## SCREEN OUTPUT

### WINDOWS

The size and shape of characters and the way in which they can be output are factors that have been greatly improved by *YS MegaBasic.* It's now possible, for instance, to confine output to a particular area of the screen; this area can be of any size and, as you'll have seen when loading, it can even cover the whole screen. These areas are referred to as 'windows', each window having a number. You can have up to ten windows on the MegaSpectrum — numbered zero to nine — and all can be utilised in your programs (although, as mentioned earlier, windows zero to three are already being used by the system).

When you run your program, all PRINT statements use window two automatically. However, it's possible to switch to another window, using the command 'CURRENT_'. The command word is followed by a single numeric expression — the number of the window to be used for all subsequent output. If the expression results in a value greater than nine then you'll get the "Illegal window" error message up on-screen. It's also possible using 'CURRENT_' to manipulate three windows at once. switching between each in turn.

As mentioned earlier, a window can be any size and take up any position on the screen. The 'WINDOW_' command is used to define the size and position of the current window and it takes the form:

**WINDOW_y,x,d,w**

Where: 'y' (the line position of the top left-hand corner of window 'y') can be from zero to 23; 'x' (the column position of the top left corner of window 'x') can be from zero to 63; 'd' is the depth of the window in character lines: and 'w' is the width of the window in character columns. As *YS MegaBasic* incorporates 64-column characters, the screen can now be thought of as being 64 character columns across and 24 character lines down.

The standard Spectrum functions ATTR and SCREEN$ still use the old co-ordinate system, but PRINT AT uses the new system. Also, the PRINT AT co-ordinates are relative to the top lefl-hand corner of the current window, whereas ATTR and SCREEN$ use absolute screen co-ordinates.

Thus, if 'y+d' is greater than 24 or 'x+w' is more than 64, you'll get the error message "Window too large"; if the values of 'd' or 'w' are less than zero, you can expect the "Window too small" error message. After a 'WINDOW_' command has been executed, the print position is reset to the top left-hand comer of the window.

## CLS AND CLW

The standard command CLS is used to clear the whole screen — which isn't much use if you just want to clear a single window. Therefore, a new command 'CLW_' is provided to perform this task. The command word is followed by one or two numbers. If two are supplied, then the first is the number of the window to be cleared and the second shows what type of 'clear' is required; if only one number is supplied then the current window is used. There are four different types of action available from the 'CLW_' command (where 'n' is the window number):

| | |
|---|---|
| **CLW_n, 0** | Clears a window as normal using the current permanent colours. |
| **CLW_n,1** | Clears a window as normal using the current permanent colours, but clears with INK instead of PAPER. |
| **CLW_n,2** | Inverts the contents of the window This option changes INK to PAPER and PAPER to INK. |
| **CLW_n,3** | Clears the attributes only. This option allows the user to change the colours of a window, without destroying its contents. |

In all cases the print position is reset to the top left-hand comer of the window.

The 'CLW_' command always uses the current permanent attributes. So, for example, if the command 'CLW_0,0' was executed

and window three was the current window, then window zero would be cleared using the attributes of window three.

## PAN AND SCROLL

It's possible, pixel by pixel, to scroll the current window up. down, left and right using the commands 'PAN_' and 'SCROLL_'. 'PAN_' scrolls a window sideways and 'SCROLL_' scrolls it vertically. Both commands must be followed by two numbers. The First indicates whether the edge of the window should be filled in with INK or PAPER; if it's a one then the edge is filled with INK — if zero, it''ll be PAPER. The second number shows in what direction and by how many pixels, the window should be scrolled. With 'PAN_' if the second number is positive then the window is scrolled left to right — if negative, right to left. With 'SCROLL_' if the number is positive the window is scrolled upwards — if negative, it goes downwards.

It's also possible to make the contents of a window wrap around. This is done using 'PANW_' and 'SCROLLW_'. These two commands are followed by a single number, specifying the direction and number of pixels to be scrolled — as with 'PAN_' and 'SCROLL_'. 'PAN_', 'PANW_', 'SCROLL_' and 'SCROLLW_' only affect the display file.

The 'FX_' command is used to control the way the *YS MegaBasic* system operates, and a number of 'FX' calls have been set aside to control choice of windows for particular tasks. The 'FX_' command is followed by two numbers, the first being the call required and the second providing any data which the call in question may need. The 'FX_' calls relating to windows are:

| | |
|---|---|
| **FX_0,n** | Sets the window used for user input and error messages. |
| **FX_1,n** | Sets the window used for automatic listings. |
| **FX_2,n** | Sets the window used for user output. |
| **FX_3,n** | Sets the window used by the front-panel. |

Try typing in some example commands to test the theory. For example, 'FX_0,5' will cause the system to use window five to display user input and error messages. And. if you've ever wanted a display like that of the Sinclair QL, then enter and run this short program:

```
10 BORDER 0
20 CURRENT_0:WINDUW_0,0,15,32
30 PAPER 2:INK7:CLW_0
40 CURRENT_1:WINDOW_0,32,22,32
50 PAPER 1:INK7:CLW_0
```

```
60 CURRENT_2:WINDOW_15,0,7,32
70 PAPER 7:INK1:CLW_0
```

You can stop any window scrolling at any time, by pressing the 'M' key.

## MODE

Characters can now be printed on the screen in four different sizes — using the 'MODE_' command:

| | |
|---|---|
| **MODE_n,1** | Calls up the 64 column character set where each character is four by eight pixels. This size doubles the maximum horizontal character resolution. |
| **MODE_n,2** | Calls up the standard size character set where each character is eight by eight pixels, These are the characters you'll find on the standard Spectrum. |
| **MODE_n,3** | Calls up the double height character set where each character is eight by 16 pixels. |
| **MODE_n,4** | Calls up the double height/double width character set, where each character is 16 by 16 pixels. This character size allows you to make use of 'stipples'. |

The 'MODE_' command can be followed by one or two numbers, in the same way as 'CLW_'; if two numbers are supplied then the first again specifies the window to be used and the second (between one and four) selects the character sizes. Each window can have its own mode and different sized characters can be mixed in the same window.

When utilising Mode 4, it's possible to introduce a feature called 'stippling', which allows characters to appear shaded. Remember that the characters of Mode 4 are four times the standard size and, as a result, each pixel in the character takes up four pixels on-screen. Thus, using stipples, it's possible to define the pattern of the four dots and to produce a shading effect. The type of pattern used is defined by the command 'STIPPLE_'. The command word is followed by a single numeric expression that indicates the pattern required. Values for the numeric expression should fall between zero and 15, the higher the value the darker the shading; a value of 15 produces solid characters, whereas zero results in the 'printing' of blank spaces.

## FONT

The MegaSpectrum has three character sets, as compared to the standard Spectrum's one. Selection of the character set is effected by 'FONT_', and this command requires a single numeric expression to define whichever one has been chosen. Here's what you get when you

use the expression with values between zero and two:

| | |
|---|---|
| **FONT_0** | Calls up the standard Spectrum character set we all know and love. |
| **FONT_1** | Calls up a character set similar to that used on the BBC Micro and Acorn Electron. |
| **FONT_2** | Calls up a character set similar to that used on the Amstrad CPC 464. |

The 'FONT_' command alters the character set used for all printing; it doesn't *just* refer to the current window.

The two new character sets are stored in RAM and thus can be altered by the user if so desired. Each character uses up eight bytes to define its shape (in much the same way as a user-defined graphic). The RAM character sets store the characters from CHR$32 up to CHR$127, the pseudo-BBC Micro character set starts at address 48000, and the pseudo-Amstrad character set starts at address 45000.

If 'a$' is the character to be defined, then the start of the eight bytes for that character is given by:

```
S+8*(CODEa$-32)
```

Where 'S' is the start of the required character set.

## CHR$, VDU AND DOWN

The Spectrum's character set consists of printable characters, and others which affect the way characters are printed — called 'control' characters. A number of new control characters have been included in the MegaSpectrum's character set:

| | |
|---|---|
| **CHR$1-4** | Select the mode of the current window. |
| **CHR$7** | Inverts the character at the cursor position. |
| **CHR$24-31** | Select the window used tor output; CHR$24 selects window zero whereas CHR$31 selects window seven. |

There's also a new command called 'VDU_' that's directly equivalent to PRINT CHR$. Let's look at some examples:

```
VDU_2
```

This makes all subsequent output appear as standard size characters, whereas the following prints out 'AB':

```
VDU_65,66
```

Strings of characters can be printed down the screen of the 'DOWN_'

command. The command takes the form:

**DOWN_y,x,a$**

Where: 'y' is the line on which the first character is to be printed; 'x' is the column at which the first character is to be printed; and 'a$' is the string to be printed. If the string runs off the bottom of the window then it wraps around and re-appears at the top.

### SPRINT AND PRINTER

Moving on to the 'SPRINT_' command, it's now possible to print characters on-screen in any size required. The command takes the form:

**SPRINT_x,y,a,b,a$**

Where: 'x' and 'y' mark the pixel position of the first character to be printed; 'a' is the number of times the characters are to be magnified in the x direction; 'b' is the number of times the characters are to be magnified in the y direction; and 'a$' is the string of characters to be printed. If the string goes off the edge of the screen then it'll wrap around and re-appear on the left-hand side. ('SPRINT_' and other *YS MegaBasic* commands take the screen origin to be the top left-hand corner of the screen, rather than the bottom left.)

The 'PRINTER_' command is provided to allow screen output to be diverted to a peripheral, such as a printer. This command is followed by a single numeric expression. If the result of that expression is zero, the output will go to the screen; if the result is other than zero, then a user-supplied routine will be called every time a character is output to the screen. The address of this machine code routine should be stored in the two bytes starting at address 59934. The character to be printed is supplied in the accumulator and the routine is terminated by a 'RET' instruction.

## GRAPHICS

### CHANGE AND SWAP

*YS MegaBasic* includes a number of commands which allow the user to directly manipulate the attributes file. 'CHANGE_' is a new command that permits alteration of certain parts of each attribute byte, while still leaving the others intact. The command word is followed by two numeric expressions, representing the 'mask' and the data. The mask, which is used to show the bits in each attribute byte that are to be changed, is best thought of in its binary form — if a bit in the mask is one, then the corresponding bit in the attribute byte is to be changed. The data shows the changes to be made. This procedure is accomplished by the following steps:

| | |
|---|---|
| **CHANGE_1** | The mask is negated — all ones are turned to zeros and all zeros are turned to ones. |
| **CHANGE_2** | Each byte in the attributes file is ANDed with the negated mask. |
| **CHANGE_3** | Each attribute byte is ORed with the data byte. |

The 'SWAP_' command allows one attribute to be swopped for another. It's followed by two numeric expressions, first the new attribute and second the old. The aittribute file is searched and if an 'old' attribute is found then it's replaced by the 'new'.

## FADE

The 'FADE_' command can produce some very spectacular effects. Enter and run this short program:

```
10 FOR A=0 TO 703
20 POKE 22528+A,PEEK A
30 NEXT A
40 FADE_0
```

The first three lines just fill the attributes file with random characters...the last line is the important one!
The command 'FADE_' is followed by a single numeric expression. The attribute file is searched and any bytes that are equal to this expression are left alone; any that aren't are decremented. This process continues until every byte in the attributes file is equal to the numeric expression.

## INVERT

The 'INVERT' command is very simple in that all it does is to invert the whole screen. It scans through each byte in the display file and negates each bit, causing INK to turn to PAPER and PAPER to INK.

## DEFG

The way in which user-defined graphics are defined on the standard Spectrum is very cumbersome...so *YS MegaBasic* provides the 'DEFG_' command. This requires a single string expression, followed by eight numeric expressions separated by commas. The string shows which graphic is to be defined ('a' to 'u') and the eight numeric expressions show the shape the graphic is to take — the first number being the top row of the character and the last, the bottom.

## GET AND PUT

It's now possible to store a portion or the screen in memory and then put it back on the screen in a different position. The two commands which accomplish this amazing feat are 'GET_' and 'PUT_'. 'GET_' saves the screen to memory, and 'PUT_' outputs the memory's contents on to the screen. 'GET_' takes the form:

**GET_0,a,y,x,d,w**

Where: 'a' is the address at which to start storing the screen information; 'y' is the line number of the top left-hand corner of the area to be saved; 'x' is the column number of the top left-hand corner of the area to be saved; 'd' is the depth of area to be saved (measured in rows of eight pixels, one standard size character square); and 'w' is the width of the area to be saved (again in groups of eight pixels).

'PUT_' and 'GET_' use the same co-ordinate system as 'ATTR' and 'SCREEN$'. Their co-ordinates are absolute, and not relative to the top left-hand side of the current window — as with 'PRINT AT'.

The 'GET_' command stores the display file information in memory followed by the attribute information. Use the 'CLEAR' command to reserve a specific piece of memory for storing screen information. You can work out the number of bytes taken by one screen area using the equation:

**9*w*d**

The 'PUT_' command is the direct opposite of 'GET_' and takes the form:

**PUT_f,a,v,x,d,w**

Where: 'f' shows the way the stored Information is to be put back on to the screen:

| | |
|---|---|
| **f=0** | Screen is overwritten by memory. |
| **f=1** | Screen is ORed with memory. |
| **f=2** | Screen is XORed with memory. |
| **f=4** | Same as 'f=0' but the current attributes are used instead of stored attributes. |
| **f=5** | Same as 'f=1' but the current attributes are used. |
| **f=6** | Same as 'f=2' but Ihe current attributes are used. |

Variables 'a', 'y', 'x', 'd' and 'w' are the same as those defined for the 'GET_' command.

### SPUT

'SPUT_' is a variation on the 'PUT_' command in that you can not only put information back on to the screen, but you can also enlarge it. 'SPUT_' takes the form:

**SPUT_a,x,y,b,c,w,d**

Where: 'a' is the address in memory of the start of the block; 'x' and 'y' are the pixel co-ordinates of the top left-hand corner of where the block is to be plotted ('SPUT_' uses the same co-ordinate system as 'SPRINT_'); 'b' is the number of times the block should be magnified in the x direction; 'c' is the number of times the block should be magnified in the y direction; 'w' is the width of the block in groups of eight pixels (as with 'GET_'); and 'd' is the depth of the block in pixels; the value should be eight times that used by 'GET_' and 'PUT_'.

'SPUT_' always uses the current attributes when placing a block on the screen; it therefore doesn't use the attributes stored in memory with the display tile information.

# CONTROLLING PROGRAM FLOW

## PROCEDURES

*YS MegaBasic* now provides the Spectrum with procedures . . . that is, subroutines which can be called just by executing their names — as if they were new commands. And, in the same way as commands, a subroutine's name can be followed by a series of expressions, if so required. These expressions can be assigned to variables at the start, ready for manipulation by the rest of the subroutine. The only drawback with *YS MegaBasic* procedures is that 'local' variables cannot be used.

Using procedures, it's possible to split a program up into a number of separate tasks. One procedure being written for each task. Each procedure can then be tested on its own and, finally, they can all be put together to form the finished program. Choose sensible names for your procedures and you'll find it much easier to follow the program flow.

The start of a procedure is defined by an '@' symbol, followed by the procedure's name. The 'define procedure' statement must be the first statement on a line; if parameters are to be passed into a subroutine then the 'define statement' must have an underline character followed by the required parameters, after the procedure's name. The parameters should be separated by commas.

The end of the procedure is defined by the 'ENDPROC_' command. Once this command is encountered, the computer will resume processing at the statement after the procedure call. To aid

legibility it is possible to tag the procedure name on to the end of the 'ENDPROC_' command. Let's look at an example:

```
9000 @DISPLAY_A,A$
9010 PAPER A:INK 9
9020 MODE_4:STIPPLE_6
9030 PRINT A$
9040 ENDPROC_DISPLAY
```

This is a simple procedure. Line 9000 defines the 'DISPLAY' procedure, and indicates that a numerical expression (A) followed by a string expression (A$) are required. Line 9010 sets the required colours and line 9020 sets the correct character size. Line 9030 prints the string which has been supplied to the subroutine and, finally, line 9040 defines the end of the procedure. To activate the procedure you'd use a line like 'DISPLAY_2,"MEGABASIC"' elsewhere in your program.

Since there are no local variables, you must make sure that those used in one procedure don't clash with others in another part of the program. Procedures can only be called from *within* a program, and so can't be used as direct commands.

## REPEAT-UNTIL

As well as procedures, *YS MegaBasic* provides 'REPEAT-UNTIL' loops. The 'REPEAT' command is used to mark the beginning of a loop and 'UNTIL_' is followed by a single numeric expression. If this numerical expression returns a zero value then the program flow jumps back to the statement after the last REPEAT instruction; if the expression is a positive number, program flow continues with the next statement as it should. 'REPEAT-UNTIL' loops can be nested up to ten deep.

## THE PROCEDURE/REPEAT-UNTIL STACK

A stack is used by procedures and 'REPEAT-UNTIL' loops to store line and statement numbers. When a procedure's called, the line and statement number of the statement after the call is placed on to the stack — this lets the MegaSpectrum know where to return to at the end of the procedure. When a 'REPEAT' command is executed, the command and line number of the statement after the 'REPEAT' command, is pushed on to the stack; this tells the 'UNTIL_' command where the last 'REPEAT' command was.

When 'ENDPROC' is executed, a jump is made to the statement stored on the stack and its value is removed. Also, if the expression of an 'UNTIL' command is true, the top value is removed from the stack. If the stack is empty but an attempt is made to remove a value, then the "PROC stack underflow" error message is displayed on-screen. The

stack can store up to ten values in all...any attempt to input an eleventh value will resull in a “PROC stack overflow” error message.

### POP AND PUSH

The ‘POP_’ command will remove a value from the stack, whereas, ‘PUSH_’ *places* a value on the stack. The ‘PUSH_’ command is followed by two numeric expressions — the statement number followed by the line number.

Anyone incorporating procedures or ‘REPEAT-UNTIL’ loops in a *YS MegaBasic* program should use the ‘PCLEAR’ command to reset the stack at the beginning of the program.

### BRANCH

The ‘BRANCH_’ command makes the computer execute a subroutine at the end of every program line. The command is followed by a single numeric expression pointing to the start of the subroutine. If its value is zero, then the MegaSpectrum acts as normal at the end of each line. The end of the subroutine is defined by ‘ENDPROC_’.

### MTASK

A simple kind of multi-tasking is available with *YS MegaBasic.* The ‘MTASK_’ command allows the program to be run from two separate places at once. Think of the program as being split into two parts: the first continues after the ‘MTASK_’ command and the second starts at the line specified by the single numeric expression following ‘MTASK_’. Once multi-tasking has been activated, the MegaSpectrum executes a line alternately from each part of the program. If the numeric expression you choose is zero, then multi-tasking is disabled; also if you enable multi-tasking, then branching is automatically disabled.
When using ‘MTASK_’, any commands which make use of the ZX Interface 1 unit must be followed by ‘!’. The program will also stop when either part finishes. If it’s likely that one part might finish before the other, then the last line should be in the form:

```
100 GO TO 100
```

This will ‘suspend’ it until the other part catches up.

## PROGRAM DEBUGGING AND EDITING

### BRANCH, TRON, TROFF AND SPEED

The ‘BRANCH_’ command will come in very useful when you’re debugging programs. If, for example, you’re interested in a particular variable, you could use a ‘BRANCH’ subroutine to print its value during the course of program execution.

Another method of checking program flow is to have the current

line number printed up on-screen via the command 'TRON'; this displays the current line number in the bottom left-hand corner of the screen. The 'TROFF' command stops the line number from being printed in the corner of the screen.

Associated with 'TRON' and 'TROFF' is the command 'SPEED_', which requires a single numeric expression to define the speed of the program run. 'SPEED_0' makes the program execute at full speed, whereas any number up to a maximum of 254 makes the execution progressively more slow. If 'SPEED_255' is used then the computer will stop after every statement and wait for the user to press a key.

## AUTO AND DELETE

When entering large programs, it's useful if the next line number can be produced every time you press Enter. This can be done by using the command 'AUTO_'. The command requires two numeric expressions separated by a comma; Ihe first is the start line number and the second, the number which is to be added to the line number every time Enter is pressed. To stop auto line numbering, use Extended Symbol Shift 'L'.

There are many occasions when it's necessary to delete a large block of program lines. This task can be accomplished using the 'DELETE_' command. Like 'AUTO_', 'DELETE_' is followed by two numeric expressions — the lirst line number of the block to be deleted and that of the last line to be deleted.

## BRON AND BROFF

Those wanting to protect their programs from prying eyes will find it useful to be able to disable the Break key This can be done using the 'BROFF' command — the Break key is re-enabled with 'BRON'. The Break key will still function during input or output operations.

Many Basics have the 'ON ERROR GOTO' command, allowing users to interrupt error conditions during program flow. The equivalent in *YS MegaBasic* is 'RESTART_', and it requires a single numeric expression indicating the line to be jumped to when an error occurs. If the 'RESTART_OFF' command is executed the Spectrum will act normally when an error occurs. 'RESTART_' will not trap either Interface 1 or new *YS MegaBasic* errors. When an error is trapped by the 'RESTART_' command, some useful information is stored in the following locations in memory:

| | |
|---|---|
| **Address 59873/4** | **The line at which the error occurred.** |
| **Address 59875** | **The statement within the line where the error occurred.** |
| **Address 59862** | **The code of the error that's occurred.** |

# SOUND

*YS MegaBasic* provides two new ways in which you can produce sound on your Spectrum — the 'PLAY_' command and the interrupt sound generator (ISG). 'PLAY_' takes the form:

**PLAY_n,l,s,d,f**

Where: 'n' produces pure notes (n=0) or white noise (n=1); 'l' is the length of each step; 's' is the start frequency; 'd' is the number of steps; and 'f' is the change of frequency alter each step. Variables 'd' and 'f' may be repeated more than once if required.

## INTERRUPT SOUND GENERATOR

Whereas 'PLAY_' is only a glorified BEEP command, *YS MegaBasic* provides a new feature, allowing sound to be produced while a program is still running. This is made possible by the inierrupt sound generator (ISG). Fifty times a second the normal program flow of the Spectrum is interrupted and a jump is made to a small machine code routine that scans the keyboard. On the MegaSpecirum, as well as scanning the Keyboard, the machine code routine has been customised to produce sounds. *YS MegaBasic* provides a number of commands to handle the ISG and the sound buffer, and these are as follows:

| | |
|---|---|
| **SOFF** | Disables the ISG — any sound being made ceases immediately |
| **SON** | Enables the ISG, so that it will resume whatever it was doing before being disabled. |
| **SREP_n** | Requires a single numeric expression (n). If the result of the expression is zero, then the data stored in the sound buffer will only be played once — if the result is one. the sound buffer's contents will be repeated for ever. |
| **SOUND_** | While quite complex, this is the main command used to manipulate the sound buffer. It takes the form:<br>**SOUND_n,a,b,c,d**<br>Where: 'n' clears the sound buffer before replacing it with something new (n=0) or adds sound on to the end of the sound buffer (n=1); 'a' plays a pure sound (a=0) or white noise (a=1); 'b' is the value to be added to the frequency after each step; 'c' is the number of steps in sequence: and 'd' is the number of times the sequence is to be repeated. |

When the ‘PLAY_’ command is executed, the ISG is disabled automatically. Therefore, before any sound can be made, you must use ‘SON’ to re-enable the ISG (note, by the way, that the ISG does not function when the line-editor is operating). The more complex the sounds get, the slower your program will run. Here’s a short example program to prove that the ISG really works:

```
10 SOUND_0,0,1,20,255
20 SREP_1
30 SON
40 MODE_4:STIPPLE_6:FONT_2
50 VDU_(128*RND*15)
60 PAPER RND*7:INK 9:
70 GO TO 50
```

Lines 10 to 30 set up the sound and lines 40 to 70 go off to perform another task.

# MACHINE CODE AND YS MEGABASIC

## DOKE AND CALL

It’s still possible to run Z80 machine code programs alongside *YS MegaBasic,* but there are one or two points worth noting as you make the transition to the MegaSpectrum.

The top section of memory above address 45000 is used by *YS MegaBasic,* thus, any machine code routines of your own that use this area will have to be moved down.

*YS MegaBasic* uses interrupt mode 2, so any routines that mess about with the interrupts will probably need some attention before they’ll work properly.

When you’re dealing with the Z80 processor it’s often necessary to POKE two-byte values into memory. *YS MegaBasic* provides the ‘DOKE_’ command to perform this task. This is followed by two numeric expressions (separated by commas); the first is the address to be POKEd and the second, the data that’s actually going to be placed into memory. The data is stored using the standard Z80 convention — that is, LSB first, MSB second.

The ‘CALL_’ command is used to call machine code routines in memory and it too is followed by a numeric expression — the address of the routine to be called. This address can then, if required, be followed by any number of numeric expressions. These expressions are evaluated and the result pushed on to the machine stack. This feature allows parameters to be passed to machine code routines.

## THE FRONT-PANEL

The 'front-panel' is provided to allow users the opportunity to alter memory and registers when using hexadecimal. The front-panel is activated either by executing the 'MON' command, or typing 'Conlrol F' while a program is running. The front-panel uses window three and this should always be at least 40 columns by 20 lines or the display won't function correctly.

Once the front-panel is activated, you'll be greeted by columns of hexadecimal numbers. On the left (in cyan) is the list of registers with the accumulator at the top; the flags register is shown in binary. The asterisk indicates the current register. On the right is the current piece of memory with the current location shown on the inverse strip; at the bottom is the input line (shown in green).

The front-panel is controlled by a number of single-letter commands; these may be followed by up to three Hex numbers. All Hex numbers must be typed out in full; if a 16-bit Hex number is required then a four digit number *must* be typed. Note that 'n' represents an eight-bit Hex number and 'nn' represents a 16-bit Hex number.

| | |
|---|---|
| **Space** | Returns to Basic. |
| **R nn** | Loads the current register with a 16-bit Hex number. |
| **P** | Advances the current register pointer by one byte. |
| **L nn nn nn** | Moves a block of memory. The first number is the start of the block, the second is the address where the block is to be moved to, and the third is the length of the block. |
| **M nn** | Sets the address of the current memory location. |
| **S** | Sets a break-point at the current memory location. |
| **K** | Continues program execution past a break-point. |
| **U** | Restores a break-point. |
| **I nn nn n** | Fills a block of memory. The first number is the start of the block, the second is the length of the block, and the third is the number with which the block is to be filled. |
| **J nn** | Calls the machine code roulineat a given address (nn). |
| **Enter** | Advances the current memory location by one byte. |
| '—' | Steps the current memory location back by one byte. |

If a two-digit Hex number is typed at the beginning of the input line, then it's entered into memory at the current location and the current location is advanced by one byte.

# SPRITE DESIGNER

## MEGASPECTRUM SPRITES

Sprites are graphic shapes that can be made to move around the screen. Their shape, colour, direction and speed can be defined by the user, and each sprite can have more than one image associated with it. As the sprite moves along it can be made to change shape by switching its image — in this way sprites can be animated. On the MegaSpectrum you can make use of eight sprites, numbered from zero to seven.

The 'SPHON_' command is used to activate the sprites. It's followed by two numeric expressions — the first showing which sprite is to be activated and the second, how the sprite should be plotted on the screen (a result of one causes it to be ORed on to the screen, two will XOR it). 'SPROFF_' deactivates the sprites and is followed by a single numeric expression — the number of the sprite to be deactivated.

Sprites on the MegaSpectrum are made up on a 16 by 16 pixel square, and each group of eight by eight pixels can have its own attribute. Two separate areas of memory are used to define sprites. The first, from address 56750 to 56893, contains all information associated with a sprite, except for its shape. Eighteen bytes are used for each sprite so the information for sprite zero starts at address 56750 and the information for sprite one starts at address 56768, and so on. The start of a particular sprite's information is given by:

**s=56750+1B*n**

Where 's' is the start of the information and 'n' is the number of the sprite required.

The 'mode' of the sprite is defined by 's+0'. Depending on the value of 's', the sprite is ORed onto the screen (s=1), the sprite is XORed onto the screen (s=2), or the sprite is inactive (s=0) — that is, it's not plotted on to the screen. The end of the sprite information area is signalled by a byte of 255; a good way to turn all sprites off at once is to execute the command 'POKE 56750.255'.

Sprites use the same co-ordinate system as 'SPUT_' and 'SPRINT_'. Thus:

| | |
|---|---|
| **s+1** | x co-ordinate of sprite (0-255). |
| **s+2** | y co-ordinate of sprite (0-175). |
| **s+3** | x increment. |
| **s+4** | y increment. |
| **s+5** | Time taken to move the sprite. |
| **s+7** | Number of images. |

| | |
|---|---|
| **s+9** | Time between the change of images. |
| **s+11/s+12** | Address of the first image. |
| **s+15** | Attribute used for erasing the sprite. |

The second area used for the image information for all sprites is between RAMTOP and address 44999, and is variable in length. A total of 36 bytes are used for each image; the first 32 bytes define the shape of the image and the last four contain the image's attributes.

As mentioned earlier, 's+11' and 's+12' show the address of the first image of each sprite. If a sprite has more than one image associated with it then it must follow the image that's pointed to by 's+11' and 's+12'. Each sprite does not have to have its own image information. If there are a number of sprites of the same shape then there only has to be one image; all that's needed is for the values 's+11' and 's+12' (for each sprite) to point to the same image.

## SPRITE DESIGNER —THE PROGRAM

The *Sprite Designer* program follows *YS MegaBasic* on tape and is loaded by:

**LOAD "SPD"**

It will run automatically, and enable sprites to be defined easily and Stored away on tape or Microdnve ready for exploitation in your own programs. The screen layout is as follows. The cyan square to the left is the area of the screen used to design images, and the area to the right is where most information is displayed. The bottom area of the screen is where user input appears.

Once it's loaded, there are a number of options to choose from:

| | |
|---|---|
| **Create a sprite** | Allows the user to define the co-ordinates, direction and speed of a sprite. |
| **View a series of images** | Provides the option to view an image on-screen at its normal size. This facility is only available if there are already some images in memory. |
| **Create an image** | Clears 36 bytes at RAMTOP to accommodate another image. |
| **Edit an image** | Allows an image to be drawn on-screen. |
| **Save images and sprites** | Saves the current sprite and image information to tape or Microdrive. |
| **Load images and sprites** | Allows the set of sprites and images to be loaded into memory. |

| | |
|---|---|
| **Copy to Microdrive** | Enables *Sprite Designer to* be transferred to Microdrive. |
| **Return to Basic** | Returns the user to *YS MegaBasic.* To re-start the *Sprite Designer*, type 'GO TO 5' and press Enter. Remember that although images are numbered from zero upwards, the first image (number zero) is highest in memory. |

You're asked to provide a file name up to eight characters in length. Three files are saved — if, say, 'Winston.n' was the filename, then the saved files would be called 'Winston.n', 'Winston.i' and 'Winston.s'. 'Winston.n' is a one-byte file indicating the number of images saved. 'Winston.i' holds the image data and 'Winston.s' provides the sprite information.

To load a set of images and sprites into your own program, you could use the following sequence:

```
9000 LOAD NS+".n"CODE 23296
9010 CLEAR 44999-36*PEEK 23296
9020 LOAD N$+".i"CODE
9030 LOAD N$+".s"CODE
```

## THE NEW COMMANDS

You'll find here a brief description of every new command in *YS MegaBasic*. A single letter is used to represent a numeric expression and a single letter followed by a dollar sign ($) represents a string expression. Check the given page for a full explanation of the syntax offered.

| | |
|---|---|
| **AUTO_a,b** | This command causes the MegaSpectrum to produce a line number every time the user presses Enter in immediate mode. To stop the computer producing line numbers, press the equals ('=') key in Extended mode and then press Enter. |
| **BACKUP** | *(WARNING: this command destroys any program already in memory.)*<br>'BACKUP' is used to copy tape files. When you execute the 'BACKUP' command you'll get the "Start tape" message on-screen and, as soon as the MegaSpeclrum detects a signal from the EAR socket, it'll display the "OK searching" message and load the next file from tape (the file type doesn't matter). Once the file has loaded you'll get the "Start tape, then press any key" message; when you're ready to save the file back to tape, press a key and the file will be re-saved. That done, the MegaSpectrum will ask if you want another copy — press the 'Y' key if you do (and the 'N' key if you don't), and the MegaSpectrum will search for the next file on tape. You can break out of this command while the MegaSpectrum is saving or loading in the normal way, but do note that if a file is longer than about 20K, the system may crash. Always execute the NEW command after 'BACKUP' has been executed. |
| **BRANCH_n** | This instruction causes a subroutine to be called at the end of every program line. The subroutine is terminated by 'ENDPROC_'. |
| **BROFF** | Enter the 'BROFF' command and you'll find that the Break key has been disabled. Note that the Break key will still function during input or output operations. |
| **BRON** | Use this to re-enable the Break key after you've used the 'BROFF' command. |

| | |
|---|---|
| **CALL_a,(n,n....)** | This calls a machine code routine at address 'a'. The address may be lfollowed by a number of numeric expressions. Each of the expressions is evaluated and the result pushed on to the machine stack; the last expression will be the stack's top value. The machine code routine removes all values from the stack and should be terminated by a Z80 'RET' instruction. |
| **CHANGE_ a,b** | This command is used to manipulate the attributes file, where 'a' represents the mask and 'b' the data. The mask is negated with each attribute byte; the data is then ORed with each attribute byte. |
| **CLW_(n),h** | Use this instruction to clear windows. If two numbers are supplied, then the first is that of the window to be cleared; where only one number is supplied then the current window is used. |
| **CURRENT_n** | This makes window 'n' the 'current' window — that is, all subsequent PRINT statements will now use this window. |
| **DEFG_a$,n...** | The 'DEFG_' command allows you to define a user-defined graphic, where a$ shows which graphic is to be defined and up to eight numbers (n,n,etc) represent the required binary pattern. |
| **DELETE_a,b** | You'd use this command to delete a block of program lines, where 'a' is the first line number of the block and 'b' the last. If the value of 'a' is greater than 'b' then the error message "B Integer out of range" is displayed on-screen. |
| **DOKE_a,b** | This command represents a two-byte POKE that's equivalent to:<br>POKE a,b-256*INT(b/256)<br>POKE a+1,INT(b/256) |
| **DOWN_y,x,a$** | This instruction prints a string downwards in the current window. If a string goes off the bottom of a window then it wraps around to the top. |
| **EDIT_n** | Use this to display line 'n' in the input line and activate the line-editor. |
| **ENDPROC_n** | This defines the end of a procedure. The name of the procedure (n) can be tagged on to the end of the command. |

| | |
|---|---|
| **EXAMINE** | This command displays the type, name, length and start of any files found on tape. To escape from it, use the Break key in the normal way. |
| **FADE_n** | This decrements each byte in the attributes file unless it's already equal to 'n'. The process is repeated until all bytes have achieved that value. |
| **FONT_n** | This instruction selects the current character set. It has no effect in Mode 1 (which uses its own special character set). |
| **FX_n,m** | This controls the on-screen operation of the YS MegaBasic system. |
| **GET_0,a,y,x,d,w** | Use this command to copy an area of the screen into memory. |
| **INVERT** | This instruction inverts the whole screen, changing INK to PAPER and PAPER to INK. |
| **KEY_n,a$** | Use this command to define a user-defined key, where 'n' is the number of the key to be defined and 'a$' is the string of characters to be assigned to the key. |
| **MODE_(n),a** | This is used to select the size of the characters for screen output. An optional number can be used to select the required window (see 'CLW_'). |
| **MON** | Use this to call up the front-panel display. It allows the user to alter and examine the memory and registers of the Z80. To return to YS MegaBasic, press the Space bar. |
| **MTASK_n** | The command which allows a YS MegaBasic program to be executed from two different places at once, offering in effect a simple form of multi-tasking. |
| **PAN_n,m** | Enables the user to scroll the current window sideways pixel by pixel; this command only scrolls the display file — the attributes are left alone. |
| **PANW_m** | This is very much like 'PAN_', but this time the contents of the window wrap around. |
| **PCLEAR** | This instruction clears the PROCedures/REPEAT-UNTIL stack and should be executed at the beginning of every program involving the use of procedures or 'REPEAT-UNTIL' loops. |

| | |
|---|---|
| **PLAY_n,l,s,d,f** | A command used to produce sound effects. Those wanting to produce tunes will find the conventional BEEP command more suitable. |
| **POP_n** | This instruction causes a value (n) from the PROCedure/REPEAT-UNTIL stack to be removed. If the command is executed when the stack is already empty, you'll get a "PROC stack underflow" error message displayed on-screen. |
| **PRINTER_n** | Use this to send output to a peripheral other than the screen — for example, a printer. |
| **PUSH_n,m** | This command pushes a value on to the PROCedure/ 'REPEAT-UNTIL' stack. If the stack's full, you'll get a "PROC stack overflow" error message on-screen. |
| **PUT_f,a,y,i,d,w** | This puts an area of memory on to the screen; it's the opposite of the 'GET_' command. |
| **REPEAT** | The 'REPEAT' command defines the beginning of a REPEAT-UNTIL loop. If the PROCedure/REPEAT-UNTIL stack is full, a "PROC slack overflow" error message flashes up on-screen. |
| **RESTART_n** | This is the YS MegaBasic equivalent of the more common 'ON ERROR GOTO' command you'll find in other Basics. 'RESTART_' will not trap errors concerning Interface 1 or new YS MegaBasic errors. |
| **RESTART_OFF** | Use this to make the MegaSpectrum act as normal when an error occurs. |
| **SCROLL_n,m** | This command scrolls the current window up or down, pixel by pixel. |
| **SCROLLW_m** | Like 'SCROLL_' but here the contents of the window are wrapped around. |
| **SOFF** | This turns off the interrupt sound generator (ISG). |
| **SON** | This turns on the interrupt sound generator (ISG). |
| **SPEED_n** | This command controls the speed of program execution when the trace function is active (see 'TRON'). |
| **SPRINT_x,y,a,b,c$** | A command that prints strings on the screen using characters whose size can be pre-delmed by the user. |

| | |
|---|---|
| **SPROFF_n,m** | Use this Instruction to turn off sprite operation. |
| **SPRON_n,m** | Use this command to turn on sprite operation. |
| **SPUT_a,x,y,b,c,w,d** | Like 'PUT_'. with this instruction you can display an area of memory on the screen. Here though, the result is magnified by factors 'b' and 'c' in the x and y directions respectively. |
| **SREP_n** | This instruction indicates if the interrupt sound generator (ISG) should repeat the sounds in the sound buffer. |
| **STIPPLE_n** | This command defines the stipple pattern to be used for Mode 4 printing. |
| **SWAP_n,m** | Use this to manipulate the attributes file; one attribute (m) is swapped for another (n). |
| **TRON** | Use 'TRON' to turn the trace function on. After the command has been executed, the current line number is displayed in the bottom left-hand corner of the screen. |
| **TROFF** | The command that turns the trace function off. |
| **UNTIL_a** | This command is used to mark the end of a 'REPEAT-UNTIL' loop. |
| **VDU_a,(,a...)** | This is directly equivalent to the command 'PRINT CHR$a'; 'a' may be repeated so that you can print more than one character at a time. |
| **WINDOW_y,x,d,w** | This command defines the size and position of the current window. |

# THE NEW ERROR MESSAGES

Eleven new error messages that'll help you learn from mistakes you make when using *YS MegaBasic.*

| | |
|---|---|
| **FX NOT IMPLEMENTED** | Linked with the 'FX_' command...you've tried to use a call which doesn't exist! |
| **ILLEGAL WINDOW** | This occurs when using 'CURRENT_' or 'FX_'; you've tried to use more than ten windows. |
| **LINE NOT FOUND** | A message that occurs when using the 'EDIT_' command...you've tried to edit a line which isn't there! |
| **MISSING PARAMETERS** | You've not provided the correct number of parameters when using a *YS MegaBasic* command. |
| **PROC STACK OVERFLOW** | You've tried to nest more than ten procedures or 'REPEAT' loops. |
| **PROC STACK UNDERFLOW** | The procedure stack is empty and you've executed 'ENDPROC_' or 'POP_'. |
| **SYNTAX ERROR** | The MegaSpectrum doesn't understand your input; this error message also occurs when the machine tries to execute a procedure definition. |
| **WINDOW TOO LARGE** | This occurs when you're using the 'WINDOW_' command, when either the window is too wide and spills off the edge of the screen, or it's too deep and tails off the bottom |
| **WINDOW TOO SMALL** | Again linked with the 'WINDOW_' command. You've made either the width or the depth of the window equal to zero. |
| **X TOO LARGE** | You've tried to plot off the side of the screen. |
| **Y TOO LARGE** | You've tried to plot off the top or bottom of the screen. |

## ORDERING YS MEGABASIC

Additional copies of *YS MegaBasic* can be ordered from *Your Spectrum.*

Send your name and address, plus cheque/Postal Order made out to *Sportscene Specialist Press Ltd,* to *YS MegaBasic Offer (extra), Your Spectrum, 14 Rathbone Place, London W1P 1DE.*
Allow 28 days for delivery — and please check that you've written your full personal details clearly.