Pegasus Basic Instructions Part 2

PLOT (mode), x, y(, x, y) - plots a point in mode# mode at coordinate x, y.

PLOT (modus),x1,y1 TO x2,y2 (TO x3,y3..)

- draws a line in mode # mode from x1, y1 to x2, y2(to x3, y3) .
PLOT TO x,y - draws a line from the last drawn point to x,y.
HLINE (mode), x1 TO x2, y - draws a horizontal line from column x1 to column x2 in row y.
VLINE (mode), x, y1 TO Y1 - draws a vertical line in column x from row y1 to row y2 .

FRAME (modus), x1, y1, x2, y2

- draws a rectangle with opposite vertices x1, y1 and x2, y2 .
BOX (mode), x1, y1, x2, y2 - draws a filled rectangle with opposite vertices x1, y1 and x2, y2 .
CIRCLE (mode), x, y, r - draws a circle with center at x, y and radius r (2-255) .
FILL x, y - fills an area around x, y.
HARDCOPY - outputs a hard copy of the graphics screen to a printer (#4).
MOVE int - moves the graphic cursor by # int steps.
MOVE TO x, y - moves the graphics cursor to x, y.
TURN angle - rotates the graphics cursor by # angle degrees.
TURN TO angle - rotates the graphics cursor to # angle degrees.
PENUP - the graphic cursor is only moved.
PENDOWN - the graphic cursor draws with every movement.
GSIZE gx, gy(, ax, ay) - sets the letter size for GPRINT in x and y direction (1-8)(and determines the spacing of the letters (0-255) to each other) .
GPRINT (mode), x, y, str$ - prints the text str$ in mode # mode from coordinate x, y. The following control characters are recognized:
RVS ON/ RVS OFF Display CTRL+1-8& CBM+1-8 Font color CTRL+ f, then color key Background CTRL+ h x-writing direction CTRL+ v y-writing direction REPEAT. . . UNTIL condition - the commands specified between REPEAT and UNTIL are executed until the condition becomes true.
WHILE condition. . . WEND - as long as the condition is true, the instructions between WHILE and WEND are executed. This loop construction, like REPEAT. . .
UNTIL, also span multiple lines.
It differs from the REPEAT loop in that the condition is tested at the beginning of the WHILE loop and not as in REPEAT. . UNTIL, at the end. So it can happen that the WHILE loop is not executed once.
IF condition THEN commands1 : ELSE commands2 - if the condition is true, commands1 are executed. If not, it is the turn of the commands2.

IF condition GOTO line1:ELSE line2

- if the condition is true, the system jumps to line1, otherwise to line2.

LIST linenr/label
RUN linenr/label
GOTO linenr/label
GOSUB linenr/label
RESTORE linenr/label

- these five commands have advanced properties. A label, ie a symbolic name, can now be used instead of a line number.
Allowed characters for the label name are the letters AZ, the digits 0-9 and the period. A label comes directly after the line number and always begins with a letter.

```
SAMPLE PROGRAM:
100 GOSUB OUTPUT
110 END
200 AUSGABE:PRINT"TEST"
210 RETURN
```

If a label begins with a command word (e.g. pie chart), the label must be marked with the keyword "LABEL":

```
SAMPLE PROGRAM:
100 GOSUB PIE GRAPH
110 END
200 LABEL PIE GRAPH
210 PRINT "CAKES TASTE GOOD!"
220 RETURN
```

Of course, calculated jumps are also possible: GOTO 1000+10* A The following should be noted: A command of the form GOSUB Z or RESTORE I*100+60000 is primarily interpreted as a jump to the label "Z" or "I".
To avoid misunderstandings, you have to put a period in front of the "Z": GOSUB . Z labels are usually searched for from the beginning of the program. However, if you write the "<" character in front of the label name, the search only starts from the current line. This can save time, but presupposes that the jump is forward, ie to higher rows.
RADIAN - switches all angles in trigonometric calculations to radians.
DEGREE - all angles are in degrees.
SWAP var1, var2 - swaps the variables var1 and var2 with each other. Both must be of the same variable type. This command corresponds to the commonly used substitute form: Q= VAR1 : VAR1= VAR2 : VAR2= Q DOKE adr, int - stores the number # int in the format LO-/ Hibyte at address # adr/ adr+1 .
DELAY int - pauses the program for # int/50s.
CLS (line) - clears the entire text screen (clears line # line) .
CLS row1, row2(, col1, col2) - clears the text screen in the range of rows row1 to row2(within columns co1 to col2) .
COLOR color register, color(, color register, color) - specifies the colors (1-16) for a given register:

1 : Font color
2 : frame colour
3 : Background color
4 : MC color 1
5 : MC color 2
6 : MC color 3
7 : Sprite-MC 1
8 : Sprite-MC 2
9 : Point color 1
10: Point color 2
11: Point color 3

MULTI ON/ OFF - switches the multicolour mode on/off.
CHAR row, column(, color)(, text) - sets the cursor to position (row, column), determines the font color and outputs a text text.
CHARSET nr - copies the original character set # nr (1/2) into the character set memory.

DEFINE CHAR bsc(, MULTI)

- determines the screen code (bsc,0-255) of the character to be defined.
If the addition MULTI is specified, a multicolor character is defined.

DEFINE SPRITE block(, MULTI)

- sets the sprite block (1-16) to change. MULTI causes an MC sprite to be defined.
CODE str$ - This command is responsible for specifying the appearance of a character or sprite.
The definition of a new character consists of eight CODE commands (corresponding to the height of a character on the screen) . 21 CODE commands are necessary for a sprite.
The length of the definition string is 8 or 24 characters, with a MULTI definition it is halved to 4 or 12 characters. The following are allowed in the definition string:

```
NORMAL MODE " ", "." Delete point
          "+", "*" punkt setzen
MULTI MODE " ", "." Delete point
          "a", "1" Punkt setzen: MC1
          "b", "2" Punkt setzen: MC2
          "c", "3" Punkt setzen: MC3
```

SPRITE nr,block,farbe,xexp,yexp,pri,mult

- sets the parameters of a sprite.
is the number of the sprite (1-8), # block is the block it is defined in (1-16), and # color is the sprite color.
Xexp,# yexp,# prior and # multi (0= off 1= on) specify the x and y magnification, priority (before or after the characters) and the multicolor mode (1= MULTI).
Within the command, parameters can be omitted without further ado, so SPRITE 1,3,7 specifies that sprite#1 is defined in block 3 and has the color 7(blue). All other parameters are not changed.
Similarly, SPRITE 7,,,1,1 determines magnification in the x and y directions only.
SPRON nr(, nr) - turns on sprite # nr.

SPRON nr1 TO nr2(, nr1 TO nr2)

- turns on sprites # nr1 to # nr2.
SPROFF nr(, nr) - turns off sprite # nr again.

SPROFF nr1 TO nr2(, nr1 TO nr2)

- turns off sprites #nr1-#nr2.
SETSPRITE nr, x, y - sets sprite # nr to coordinate x, y (x=0-511, y=0-255) .

THE NEW FUNCTIONS:
var = YOU(num)
converts #num from radians to degrees.
var = ROW(num)
converts #num from degrees to radians.

var = FRAC(num) Decimal part of number # num.
var = MOD(num1, num2) integer remainder of division num1/ num2 var = ROUND(num) rounds # num to an integer result.
var = ROUND(num, num) rounds # num to # num decimal places.
var = DEC(str$) converts the hexadecimal number str$ to the decimal system.
var = BIN(str$) converts the binary number str$ into the decimal system.
var = DEEK(addr) corresponds to PEEK(addr)+256* PEEK(addr+1) .
var = INSTR(str1$, str2$(, pos)) checks whether str2$ is contained in str1$(searching from position # pos).
var = JOY(port) reads the joystick port # port.

```
    1
      If the fire button is pressed,
  8 | 2 so the values increase by
    |   128.
7 --0-- 3
    |
  6 | 4
    5
```

var = POT(nr)
Value of analog input #nr (1-4).

var = SCREEN (row, col) Screen code of character at position # row, # col.
var = TEST (x, y) checks whether a point is set in the graph.
var = USING format$; var. :
creates a string in which the specified variables are formatted. The following conventions apply here:
is a number ### .## is a number with 3 digits before and 2 after the decimal point.

```
! is the first character of a
      Strings.
// are (number of spaces +2) characters
      a string.
& is the complete string.
<zchn character zchn is directly over-
    taken
```

If there is a + before a #, the sign is always specified.
EXAMPLES:

```
?USING "###.###";π       => "  3.1415"
?USING "###.###"-π       => "  3.1415-"
?USING "/     /";"COMMODORE" => "COMMO"
```

var$ = DS$
Error status of the current drive.

var$= Hex$(int) converts # int to a hex number.
var$= BIN$(int) converts # int to binary.
var$= SPACE$(int) produces a string with # int spaces.
var$= UCASE$(str$) converts all lowercase letters in str$ to uppercase.
var$= STRING$(asc, num) produces a string of # num characters with ASCII value # asc.
var$= STRING$(str$, num) produces a string containing num times str$.
var$= INPUT$(num) reads # num characters from the keyboard.
var$= INPUT$(# filenr) reads characters from the file # filenr until a CR is read.
Corresponds to INPUT# filenr, but var$ can also read separators (comma, semicolon).
var$= INPUT(# filenr, num) reads # num characters from the file # filenr.
var$= TIME$ returns the time (does not correspond to TI$) .

var = SPRITEX(no)
var = SPRITEY(no)
X or y coordinate of sprite #nr.
EXAMPLE OF AN EXTENSION:

An example of how PBasic can be expanded with new commands and functions is the "WINDOW.O" program. It provides commands for programming windows. You load and start it with BRUN " WINDOW.O" .
This extension is integrated into PBasic via the vectors already mentioned at $0334 and $0336.
Please load the third part of the Pegasus-Basic manual now.