FUZE BASIC®

Programmer's Reference Guide

and, a whole lot of fun…

FUZE BASIC®

Programmer's Reference Guide

CODING & COMPUTING WORKSHOPS
for Schools, Academies, Colleges, Universities,
Computing Clubs, Holiday Camps & Special Events

www.fuze.co.uk

To contact FUZE call 01844 239 432
(UK 09:30 am to 17:00 pm weekdays)

or email contact@fuze.co.uk

FUZE BASIC is developed by FUZE Technologies Ltd in the UK by;

Jon Silvera - Project manager and author of this guide
Luke Mulcahy - Lead programmer
David Silvera - FUZE Product manager
Colin Bodley - Technical consultant & Documentation contributor
Additional information – www.fuze.co.uk

Manual Version: Date: 10th May 2017 FUZE BASIC Version: 3.9.1

Introduction

Introduction & Welcome to FUZE BASIC

This reference guide aims to provide a comprehensive and detailed explanation of every command in the FUZE BASIC library.

It is an evolving document because FUZE BASIC is constantly being updated and improved. To download the latest version please visit the resources section on the www.fuze.co.uk website.

FUZE BASIC is currently available for Microsoft Windows®, Raspberry Pi, The FUZE and Linux.

Installing FUZE BASIC

For details on installing and getting started we highly recommend you begin with the FUZE BASIC Project Worksheets which can be downloaded for free from the FUZE website.

Conventions used in this guide:

There are many example programs listed in this manual. The following colours are used to determine the different program construct types;

- **COMMANDS - STATEMENTS & FUNCTIONS**
- **SYSTEM & FIXED VARIABLES**
- **USER DEFINED VARIABLES**
- REM or comments

Using FUZE BASIC with Arduino / Genuino, BBC micro:bit and the OWI USB Robot Arm.

FUZE BASIC will only recognise an Arduino or BBC micro:bit device once the driver software has been installed.

For Arduino devices: Please visit the official Arduino website at https://www.arduino.cc and download / install the latest version.

For BBC micro:bit devices: Please visit the official ARM mbed site at https://developer.mbed.org/handbook/Windows-serial-configuration and download & install the driver.

If you're using multiple devices then please see the **DETECTDEVICE** and **SETDEVICE** statements so you can easily switch between them.

OWI USB Robot Arm (Maplin): No driver is required for Linux and Raspberry Pi based systems but if you are using FUZE BASIC for Windows then a driver is required. Please see the FUZE BASIC website for more details.

Community - your input is welcomed

If you spot any errors or think we could explain something better, or even have alternative suggestions for examples then please submit them to contact@fuze.co.uk. Please note however that this intended to be a none technical guide - please keep submissions short, simple and well documented.

Many thanks from teamFUZE

Immediate mode Commands

Immediate mode commands are entered into FUZE BASIC's direct mode. For example try typing in; **PRINT** 10 + 20 + 30 and then press enter.

The following system commands can be used in immediate mode are explained in the following pages.

Please note:
You do not need to learn immediate mode commands to use BASIC. In fact you may never even need to use them. You can simply jump straight to the editor by pressing F2.

Immediate mode is the start mode for FUZE BASIC and depending which retro styled interface you have selected will look something like this;

Press the **INS** key to switch between them. Then press **F2** or type **EDIT** to switch to the code editor.

CD

Purpose

Change the current directory

Syntax

CD foldername

Description

Change the current folder to the one specified.

Associated

LOAD, SAVE, CHAIN, DIR

Example

CD fuzebasic

Switch to the folder named fuzebasic. There must be a folder called 'fuzebasic' in the current folder to work.

CD ..

Go back to the parent folder (CD ../.. will go back two folders.)

CD /

Go back to root folder.

CHAIN

Purpose

Load a program into memory and immediately execute it.

Syntax

CHAIN *filename$*

Description

Loads in a program from the local non-volatile storage. As with SAVE, you need to supply the filename without any quotes. Do not include the .fuze file extension. Note, if your filename has spaces then you must enter it within quotation marks. CHAIN "my game" for example.

The program will start automatically as soon as it loads.

Associated

LOAD, SAVE, NEW

Example

CHAIN demos
RUN

CLEAR

Purpose

Clear variable memory.

Syntax

CLEAR

Description

Clears all variables and deletes all arrays. It also removes any active sprites from the screen. Stopped programs may not be continued after a CLEAR command.

Associated

NEW

Example

```
variable=10
PRINT variable
CLEAR
PRINT variable
```

CLS

Purpose

Clear the video display screen.

Syntax

CLS

Description

Clears the video display screen and places the cursor in the top left corner. The background is set to the current background (PAPER) colour.

Associated

PAPER

Example

```
PAPER=WHITE
INK=BLACK
CLS
PRINT "Hello World"
END
```

CONT

Purpose

Continue running a program that has been stopped.

Syntax

CONT

Description

Continues program execution after a STOP instruction.
Variables are not cleared.

Associated

STOP

Example

CONT

DIR

Purpose

List FUZE BASIC program files in the current or specified directory.

Syntax

DIR [*directory*]

Description

Lists the program files in your current working directory or the optional specified *directory*. These will have the .fuze file extension.

Associated

LOAD, SAVE

Example

DIR

EDIT

Purpose

Edit the program in memory.

Syntax

EDIT

Description

Edits the program in memory using a full screen editor.

Example

EDIT

ESCAPE (key press)

Purpose

Halt program execution

Syntax

ESCAPE (key press)

Description

Halts the current program's execution and returns to immediate mode. All variables remain intact so it is possible to examine them for debugging.

Associated

Example

Press ESCAPE key when a program is running.

EXIT

Purpose

Exit FUZE BASIC and return to the environment.

Syntax

EXIT

Description

Exit FUZE BASIC and return to the environment you started it in.

Associated

Example

```
EXIT
```

HELP

Purpose

Display help commands in immediate mode

Syntax

HELP

Description

Lists the full immediate mode command set.

Associated

Example

HELP

INSERT (key press)

Purpose

Changes the GUI skin in immediate mode

Syntax

INSERT (key press)

Description

Cycles through the various retro styled skins in immediate mode. BBC Micro anyone?

Associated

Example

Press INSERT key in immediate mode to change retro skin.

LISTGAMEPADS

Purpose

To display any connected joysticks and gamepads.

Syntax

LISTGAMEPADS

Description

This lists, by name, any attached joystick devices currently connected via USB or wireless USB, if supported by the OS.

Associated

Example

LISTGAMEPADS

LOAD

Purpose

Load a program into memory.

Syntax

LOAD *filename$*

Description

Loads in a program from the local non-volatile storage. As with SAVE, you need to supply the filename without any quotes. Do not include the .fuze file extension. Note, if your filename has spaces then you must enter it within quotation marks. LOAD "my game" for example.

Associated

SAVE

Example

LOAD demos
RUN

LOOK

Purpose

Not much at all

Syntax

LOOK

Description

Erm...

Associated

Example

LOOK

NEW

Purpose

Start a new program

Syntax

NEW

Description

Deletes the program in memory. There is no verification and once it's gone, it's gone. Remember to save first!

Associated

CLEAR

Example

NEW

RUN

Purpose

Runs the program in memory.

Syntax

RUN

Description

Runs the program in memory. Note that using RUN will clear all variables.

Associated

ESCAPE (key press)

Example

RUN

SAVE

Purpose

Saves your program to the local non-volatile storage.

Syntax

SAVE *filename$*

Description

Saves your program to the local non-volatile storage. The *filename$* is the name of the file you wish to save and may not contain spaces. If you have already saved a file, then you can subsequently execute SAVE without the filename and it will overwrite the last file saved. (This is reset when you load a new program or use the NEW command)

Note: If you use spaces in a file name then you must encase the name in quotation marks i.e “my prog name”

Associated

LOAD

Example

SAVE testprog

SHOWKEYS

Purpose

List the function key definitions for immediate mode

Syntax

SHOWKEYS

Description

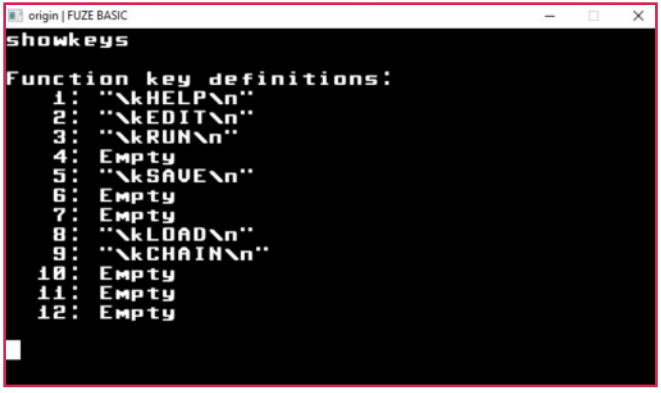
It is possible to set the 12 function keys at the top of the keyboard to user defined values. This command will show what the current definitions are. By default function key **2** is defined to enter the **EDIT** command and **F3** the **RUN** command but it is possible to overwrite these.

Associated

keyF1$, keyF2$, keyF3$, keyF4$, keyF5$, keyF6$, keyF7$, keyF8$, keyF9$, keyF10$, keyF11$, keyF12$

Example

```
REM Set key F5 to clear screen
keyF5$ = "CLS\n"
SHOWKEYS
```



TROFF

Purpose

Disables TRON

Syntax

TROFF

Description

If TRON is enabled then TROFF switches it off.

Associated

TRON

Example

TROFF

TRON

Purpose

Enables TRON mode for debugging.

Syntax

TRON

Description

Switching TRON on has a very dramatic effect. Running your program when TRON is enabled displays a section of code on screen to show exactly which line is being executed.

Associated

TROFF

Example

TRON

You can adjust **TRON** settings with the following controls;

CTRL+ UP / DOWN to change speed
CTRL+ LEFT to Pause (hold down)
CTRL+ LEFT + RIGHT to step
CTRL+ RIGHT toggle transparent
CTRL+ SHIFT + UP toggle TRON display
CTRL+ PAGEUP increase number of lines displayed
CTRL+ PAGEDOWN decrease number of lines displayed
CTRL+ SHIFT + PAGEUP move TRON display up
CTRL+ SHIFT + PAGEDOWN move TRON display down

VERSION

Purpose

Print the current version of FUZE BASIC.

Syntax

VERSION

Description

Displays the current version of FUZE BASIC along with relevant program information.

Example

```
VERSION
```

Contents

Arithmetic operators

Purpose

Arithmetic operators are used to perform calculations with numbers and variables.

Syntax

```
value = 10 * 10
answer = value DIV 10
IF value <= 10 THEN ...
```

Description

^	———	Exponent or “raise to the power of”
*****	———	Multiply
/	———	Divide
MOD	———	Modulo
DIV	———	Integer Divide
+	———	Addition
-	———	Subtraction
<	———	Less than
<=	———	Less than or equal to
>	———	More than
>=	———	More than equal to
=	———	Equal to
<>	———	Not equal to

Contents

Bitwise operators

Purpose

Binary operators allow logical operations to be performed on numbers and variables.

Syntax

value = number | number
value = number & number
value = number XOR number
value = number << bits to shift
value = number >> bits to shift

Description

|	Bitwise OR
&	Bitwise AND
XOR	Bitwise XOR
<<	Bitwise shift left
>>	Bitwise shift right

Contents

Logical operators

Purpose

Logical operators

Syntax

IF value = NOT value2 THEN...

IF value = 1 AND value2 = 2 THEN ...

IF value = 1 OR value2 = 2 THEN ...

Description

NOT	Logical NOT
AND	Logical AND
OR	Logical OR

Contents

Numerical representation

Purpose

Number values are generally entered in decimal but it can be very useful to represent a value in its binary or hexadecimal format.

Syntax

129	decimal
0b10000001	binary - prefixed with “0b”
0x81	hexadecimal - prefixed with “0x”

The command..

```
PRINT 0x81 + 0b10000001
```

..will display 258

Functions, Constants & Procedures

These are the commands and statements used to build a program. The best way to learn them is to copy the examples into the editor and run them, then change and expand them. What's the worst that could happen? Just don't go and invent a world dominating artificial intelligence that's all we ask!

Arithmetic

ABS
ACOS
ASIN
ATAN
ATAN2
CLOCK
COS
DEG
EXP
LOG
MAX
MIN
INT
PI
PI2
LERP
RAD
SGN
SIN
SPLINE
SQRT
TAN

Variables

DATA
DIM
FALSE
LOCAL
READ
RESTORE
RND
SEED
SWAP
TRUE

IF statements

CASE
DEFAULT
ELSE
ENDCASE
ENDIF
ENDSWITCH
IF THEN
SWITCH

Loops

FOR LOOP
LOOP REPEAT
UNTIL REPEAT
WHILE REPEAT

Procedures & FN

DEF FN
DEF PROC
ENDPROC
FN
PROC

Program flow

BREAK
CONTINUE
END
STOP
WAIT

Text handling

ASC
CHR$
DEFCHAR
FONTSIZE
HTAB
HVTAB
INK
LEFT$
LEN

Contents

ROTATEIMAGE
SAVEREGION
SAVESCREEN
SCALEIMAGE
SCROLLDOWN
SCROLLLEFT
SCROLLRIGHT
SCROLLUP
SETALPHA
SETIMAGEORIGIN
TRIANGLE
VLINE

Sprite graphics

ADVANCESPRITE
CLONESPRITE
GETSPRITEANGLE
GETSPRITEDISTANCE
GETSPRITEH
GETSPRITEPIXELRGBA
GETSPRITEW
GETSPRITEX
GETSPRITEY
HIDESPRITE
LOADSPRITE
NEWSPRITE
PLOTSPRITE
SETSPRITEALPHA
SETSPRITEANGLE
SETSPRITEFLIP
SETSPRITEORIGIN
SETSPRITESIZE
SPRITECENTRE
SPRITECOLLIDE
SPRITECOLLIDEPP
SPRITEOUT

Turtle graphics

LEFT
MOVE
MOVETO
PENDOWN
PENUP
RIGHT
TANGLE

Sound & Music

LOADMUSIC
LOADSAMPLE
PAUSECHAN
PAUSEMUSIC
PLAYSAMPLE
RESUMECHAN
RESUMEMUSIC
SETMUSICVOL
STOPCHAN
STOPMUSIC
TONE

File handling

CLOSE
EOF
FFWD
INPUT#
LISTDIR
OPEN
PRINT#
REWIND
SEEK

Serial IO

SCLOSE
SGET
SGET$
SOPEN
SPACE$
SPUT
SPUT$
SREADY

USB Robot arm

ARMBODY
ARMELBOW
ARMGRIPPER
ARMLIGHT
ARMRESET
ARMSHOULDER
ARMWRIST

Contents

BBC micro:bit

MBACCELX
MBACCELY
MBACCELZ
MBANALOGREAD
MBANALOGWRITE
MBBUTTONA
MBBUTTONB
MBCLS
MBCOMPASSFIELDSTRENGTH
MBCOMPASSHEADING
MBCOMPASSX
MBCOMPASSY
MBCOMPASSZ
MBDIGITALREAD
MBDIGITALWRITE
MBFACEDOWN
MBFACEUP
MBFREEFALL
MBGESTURE
MBLINE
MBPITCH
MBPLOT
MBPRINT
MBRECT
MBROLL
MBSHAKE
MBTEMP
MBTILTDOWN
MBTILTLEFT
MBTILTRIGHT
MBTILTUP
MBTOUCH

RPi senseHAT

SENSEACCELX
SENSEACCELY
SENSEACCELZ
SENSECLS
SENSECOMPASSX
SENSECOMPASSY
SENSECOMPASSZ
SENSEGETRGB
SENSEGYROX
SENSEGYROY
SENSEGYROZ
SENSEHEIGHT
SENSEHFLIP
SENSEHUMIDITY
SENSELINE
SENSEPLOT
SENSEPRESSURE
SENSERECT
SENSERGBCOLOUR
SENSESCROLL
SENSETEMPERATURE
SENSEVFLIP

Devices

DETECTDEVICES
DEVICETYPE

Digital IO

DIGITALREAD
DIGITALWRITE
PINMODE
PWMWRITE
SOFTPWMWRITE

Analog IO

ANALOGREAD
ANALOGWRITE

ABS

Purpose

Return the absolute value of the argument.

Syntax

Positivenumber = ABS(*number*)

Description

Returns the absolute value of the supplied argument *number* i.e. If the argument is negative, make it positive.

Associated

ABS, SGN

Example

```
PRINT FN ElapsedYears( 1966, 2013 )
PRINT FN ElapsedYears( 2013,1966 )
END

REM Return Number of years elapsed
REM Between two dates
DEF FN ElapsedYears( Year1, Year2 )
=ABS( Year1-Year2 )
```

ACOS

Purpose

Returns the arc cosine of the supplied argument.

Syntax

Angle = ACOS(*cosine*)

Description

This is the inverse of the COS function returning the angle for a given cosine.

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
PRINT "Angle with cosine = 0.5: "
DEG
REM 60 Degrees
PRINT "In Degrees: "; ACOS( 0.5 )
RAD
REM PI/3 Radians
PRINT "In Radians: "; ACOS( 0.5 )
CLOCK
PRINT "In Minutes: "; ACOS( 0.5 )
END
```

ADVANCESPRITE

Purpose

Advances a sprite a specified amount

Syntax

ADVANCESPRITE(*sprite*, *distance*)

Description

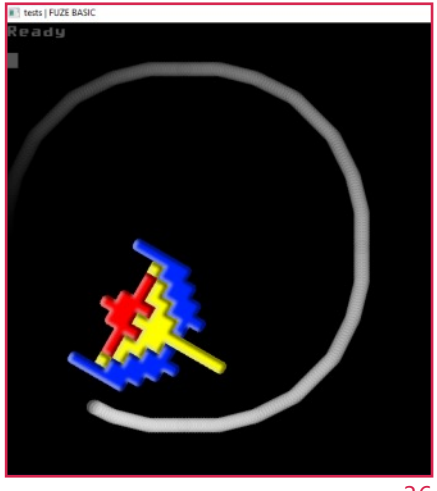
Moves a sprite forward by the specified distance. The direction is set by the SETSPRITEANGLE function. This is very useful when used with rotating sprites.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
sprite = NEWSPRITE( 1 )
LOADSPRITE( "fighter.png", sprite, 0 )
PLOTSPRITE( sprite, 0, GHEIGHT/2, 0 )
FOR angle = 0 TO 300 LOOP
SETALPHA((100 / 360)*angle)
CIRCLE (GETSPRITEX(sprite), GETSPRITEY(sprite), 10, 1)
SETSPRITEANGLE( sprite, angle )
ADVANCESPRITE( sprite, 5 )
UPDATE
REPEAT
END
```



ANALOGREAD

Purpose

Returns the value from an analog input.

Syntax

ANALOGREAD(0)

Description

Depending on the system will returns a value between 0 and 255 (FUZE) or 0 and 1023 (Arduino & micro:bit) depending on the current received in the specified input.

Note:

If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

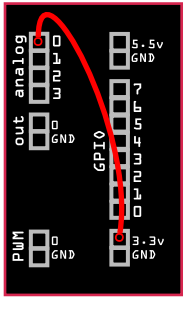
ANALOGREAD, ANALOGWRITE, DIGITALREAD, DIGITALWRITE, PINMODE, PWMWRITE, SOFTPWMWRITE

Example

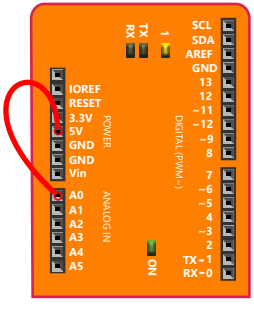
```
REM Attach 3.3V (FUZE & micro:bit) or 5V (Arduino) to analog pin 0

LOOP
PRINTAT (0, 0);  ANALOGREAD (0)
UPDATE
REPEAT
```

Connect the cable to analog pin 0. The value will adjust between high (connected - this might be 255 or 1024 depending on the device), low (connected but 0V or GND) or floating (not connected).



FUZE IO board



Arduino or compatible IO board

ANALOGWRITE

Purpose

Sends a value to an analog output.

Syntax

ANALOGWRITE(0)

Description

Depending on the system, a value between 0 and 255 (FUZE 3.3V max) (Arduino 5V max & micro:bit 3.3V max) can be sent to the pin.

Note:

If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

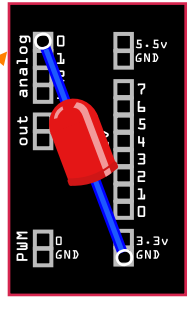
ANALOGREAD, ANALOGWRITE, DIGITALREAD, DIGITALWRITE, PINMODE, PWMWRITE, SOFTPWMWRITE

Example

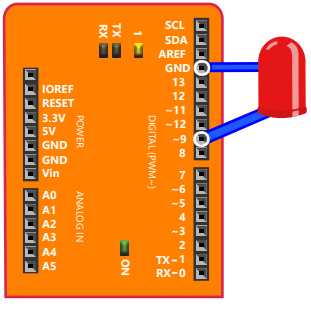
REM Attach an LED (active leg) to the analog out pin and the other leg to GND. For an Arduino device we use the PWM output pins to send an anlog signal out.

```
LOOP
  FOR a = 0 TO 255 STEP 10 LOOP
    ANALOGWRITE (9, a) // (change 9 to 0 for FUZE)
    WAIT (0.01)
  REPEAT
  FOR a = 255 TO 0 STEP -10 LOOP
    ANALOGWRITE (9, a) // (change 9 to 0 for FUZE)
    WAIT (0.01)
  REPEAT
  UPDATE
REPEAT
```

NOTE: long leg is active and goes to the power supply



FUZE IO board



Arduino or compatible IO board

ASC

Purpose

Return the ASCII code for a string character.

Syntax

asciicode=ASC(*string$*)

Description

Returns the ASCII value represented by the first character of *string$* e.g. "A" would return 65. It is the opposite of the CHR$ function.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
code=ASC( "A" )
PRINT "ASCII value of letter A is: "; code
PRINT "ASCII char of code 65: "; CHR$( code )
END
```

ASIN

Purpose

Returns the arc sine of the supplied argument.

Syntax

angle = ASIN(*sine*)

Description

This is the inverse of the SIN function returning the angle for a given sine.

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
PRINT "Angle with sine = 0.5: "
DEG
REM 30 Degrees
PRINT "In Degrees: "; ASIN( 0.5 )
RAD
REM PI/6 Radians
PRINT "In Radians: "; ASIN( 0.5 )
CLOCK
PRINT "In Minutes: "; ASIN( 0.5 )
END
```

ATAN

Purpose

Returns the arctangent of the supplied argument.

Syntax

angle=ATAN(*tangent*)

Description

This is the inverse of the TAN function returning the angle for a given tangent.

Associated

ACOS, ASIN, ATAN, ATAN2, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
PRINT "Angle with tangent equal to 1: "
DEG
REM 45 Degrees
PRINT "In Degrees: "; ATAN( 1 )
RAD
REM PI/4 Radians
PRINT "In Radians: "; ATAN( 1 )
CLOCK
PRINT "In Minutes: "; ATAN( 1 )
END
```

ATAN2

Purpose

Returns the arctangent between the point specified and the origin.

Syntax

angle=ATAN2(y, x)

Description

Unlike ATAN, ATAN2 requires two values. It returns the angle in radians between the origin and the values specified.

Associated

ACOS, ASIN, ATAN, ATAN2, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
CLS
LOOP
PRINTAT (0, 0);  "Degrees: ";  ATAN2 (MOUSEY, MOUSEX)
REPEAT
END
```

Move your mouse to see the value change.

BREAK

Purpose

Provide an early exit from a loop.

Syntax

BREAK

Description

Terminates a loop before the terminating condition is met.

Example

```
REM Loop until the space bar is pressed
PRINT "Press the space bar to continue"
LOOP
  IF INKEY = 32 THEN BREAK
REPEAT
END
```

Associated

CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT UNTIL, UNTIL REPEAT, WHILE REPEAT

CHR$

Purpose

Returns the string character for the specified ASCII code.

Syntax

*character$*=CHR$(*asciicode*)

Description

Returns a one-character string consisting of the character

corresponding to the ASCII code indicated by the *asciicode* argument. This is the opposite of the ASC function.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
PRINT "The following prints the letter A"
PRINT CHR$( 65 )
PRINT CHR$( ASC( "A" ))
PRINT "The following prints the number 3"
PRINT CHR$( 51 )
PRINT CHR$( ASC( "0" ) + 3 )
END
```

CIRCLE

Purpose

Draw a circle on the screen.

Syntax

CIRCLE(*centrex*, *centrey*, *radius*, *fill*)

Description

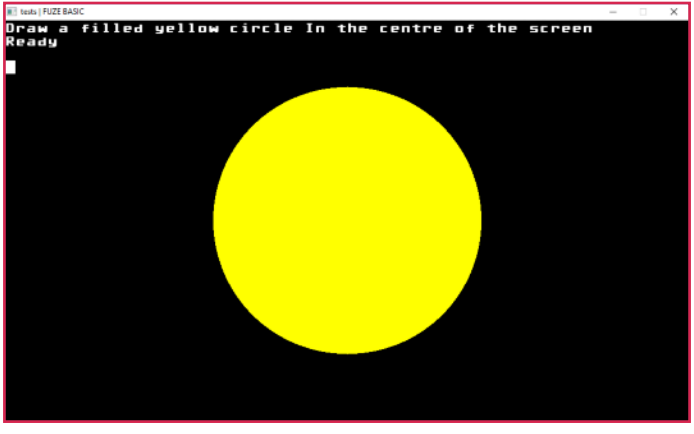
Draws a circle at position (*centrex*, *centrey*) with the specified *radius* in the current foreground COLOUR. The final parameter *fill* is either TRUE or FALSE, and specifies filled (TRUE) or outline (FALSE).

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
PRINT "Draw a filled yellow circle ";
PRINT "In the centre of the screen"
COLOUR = YELLOW
CIRCLE( GWIDTH/2, GHEIGHT/2, 200, TRUE )
UPDATE
END
```



CLEARKEYBOARD

Purpose

Clear all pending keyboard input.

Syntax

CLEARKEYBOARD

Description

Clears the keyboard input buffer.

Associated

CLEARKEYBOARD, GET, GET$, INKEY, INPUT, SCANKEYBOARD

Example

```
PRINT "Press space to continue"
WHILE NOT SCANKEYBOARD( SCANSPACE ) LOOP
REPEAT
CLEARKEYBOARD
END
```

CLOCK

Purpose

Set angle units to minutes.

Syntax

CLOCK

Description

Switches the internal angle system to clock mode. There are 60 minutes in a full circle.

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
PRINT "ATAN( 1 ) = "
DEG
PRINT ATAN( 1 ); " Degrees"
CLOCK
PRINT ATAN( 1 ); " Minutes"
RAD
PRINT ATAN( 1 ); " Radians"
END
```

CLONESPRITE

Purpose

Create a new sprite from an existing one.

Syntax

CLONESPRITE (*sprite handle*)

Description

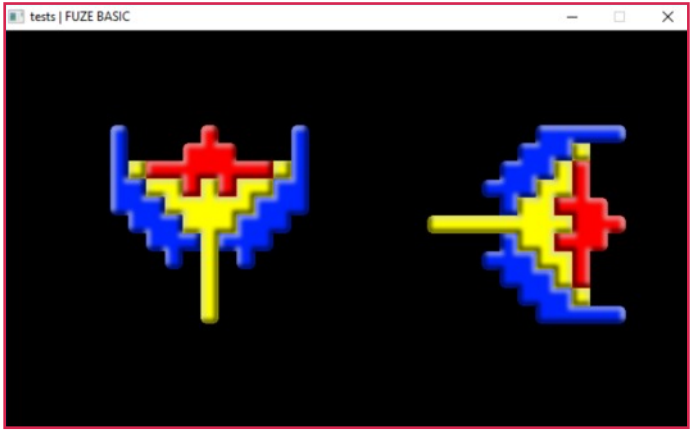
Copies a sprite and its settings from an existing one (*sprite handle*) into a new one.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
sprite1 = NEWSPRITE ( 1 )
LOADSPRITE ("fighter.png", sprite1, 0 )
sprite2 = CLONESPRITE (sprite1)
SETSPRITEANGLE(sprite2,90)
LOOP
    PLOTSPRITE ( sprite1, 100, 100, 0 )
    PLOTSPRITE ( sprite2, 400, 100, 0 )
REPEAT
```



CLOSE

Purpose

Close a file after use.

Syntax

CLOSE(*handle*)

Description

The CLOSE instruction closes the file specified by *handle* after use and makes sure all data is securely written to the storage medium.

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN( "testfile.txt" )
PRINT# handle, "Colin"
CLOSE( handle )
handle = OPEN("testfile.txt")
INPUT# handle, Name$
CLOSE( handle )
PRINT "Name: " + Name$
END
```

CLS / CLEARSCREEN

Purpose

Clear the visible screen.

Syntax

CLS

Description

Completely clear the screen of text and graphics. Note, this does not clear sprites from the screen.

Example

```
PRINT "Hello World"
WAIT ( 1 )
CLS
END
```

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

CLS2 / CLEARSCREEN2

Purpose

Clear screen without an update.

Syntax

CLS2

Description

Ideally suited to games and graphical programming. **CLS2** clears the background or buffer screen and does not issue an update command. This means you can wipe the screen buffer, redraw on it and then issue an update. This ensures flicker free updates. It is also faster than **CLS**.

Example

```
y = GHEIGHT / 2
radius = GWIDTH / 10
FOR x = 0 TO GWIDTH STEP 0.5 LOOP
CLS
   COLOUR = PINK
   CIRCLE ( x, y, radius, 1 )
   UPDATE
REPEAT
FOR x = GWIDTH TO 0 STEP -0.5 LOOP
   CLS2
   COLOUR = YELLOW
   CIRCLE ( x, y, radius, 1 )
   UPDATE
REPEAT
```

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

COLOUR

Purpose

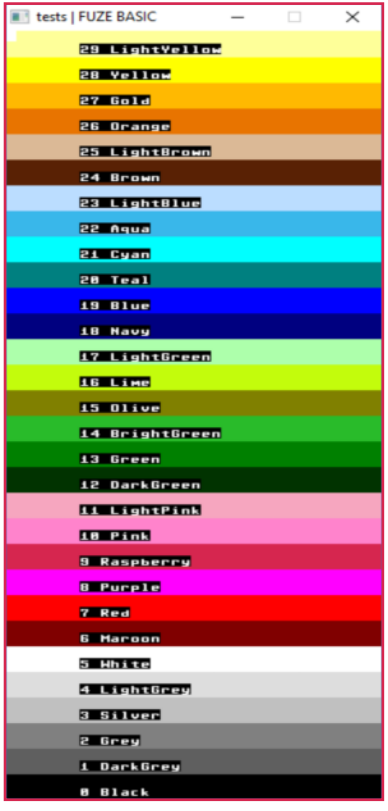
Set the current graphics plot colour.

Syntax

COLOUR = *setcolour*

Description

Set the current graphics plot colour as follows:



Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
FONTSIZE(1)
height = GHEIGHT / 30
FOR col = 0 TO 29 LOOP
COLOUR = col
RECT(0, col * height, GWIDTH, height, 1)
READ NameOfColour$
text$ = STR$(col) + " " + NameOfColour$
PLOTTEXT(text$, GWIDTH / 2, (height * col) + height / 2);
REPEAT
UPDATE
END
DATA "Black", "DarkGrey", "Grey", "Silver"
DATA "LightGrey", "White", "Maroon", "Red"
DATA "Purple", "Raspberry", "Pink", "LightPink"
DATA "DarkGreen", "Green", "BrightGreen", "Olive"
DATA "Lime", "LightGreen", "Navy", "Blue", "Teal"
DATA "Cyan", "Aqua", "LightBlue", "Brown"
DATA "LightBrown", "Orange", "Gold", "Yellow"
DATA "LightYellow"
```

CONTINUE

Purpose

Continue a loop.

Syntax

CONTINUE

Description

Will cause the loop to re-start at the line containing the LOOP instruction, continuing a FOR instruction and re-evaluating any WHILE or UNTIL instructions.

Example

```
REM Prints 1 to 10 but skips over 5
FOR num = 1 TO 10 LOOP
   IF num = 5 THEN CONTINUE
   PRINT num
REPEAT
END
```

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT, UNTIL, UNTIL REPEAT, WHILE REPEAT

COPYREGION

Purpose

Copy a region of the screen from one location to another.

Syntax

COPYREGION(*oldx*, *oldy*, *width*, *height*, *newX*, *newY*)

Description

This enables you to duplicate the contents of a section of the screen from one place to another. This could be used for example to create a background for a game by drawing an image and then duplicating it. The region to be copied is specified by the rectangle at coordinates (*oldx*,*oldy*) *width* pixels wide and *height* pixels high. The region is copied to coordinates (*newx*, *newy*)

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
COLOUR = RED
RECT( 0, 0, 50, 50, TRUE )
COLOUR = WHITE
LINE( 0, 50, 50, 50 )
LINE( 0, 25, 50, 25 )
LINE( 0, 25, 0, 50 )
LINE( 50, 25, 50, 50)
LINE( 25, 0, 25, 25 )
LINE( 0, 0, 50, 0 )
FOR X = 0 TO GWIDTH STEP 50 LOOP
  FOR Y = 0 TO GHEIGHT STEP 50 LOOP
    COPYREGION( 0, 0, 50, 50, X, Y )
  REPEAT
REPEAT
UPDATE
END
```

COS

Purpose

Returns the cosine of the given angle.

Syntax

cosine=COS(*angle*)

Description

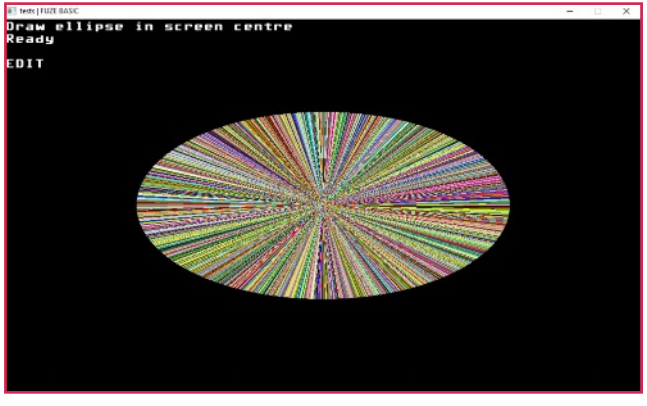
Returns the cosine of the argument *angle*. This is the ratio of the side of a right angled triangle, that is adjacent to the angle, to the hypotenuse (the longest side).

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
CLS
PRINT "Draw ellipse in screen centre"
DEG
FOR Angle = 0 TO 360 STEP 0.01 LOOP
COLOUR=RND(30)
Xpos = 300 * COS( Angle ) + GWIDTH / 2
Ypos = 150 * SIN( Angle ) + GHEIGHT / 2
LINE (GWIDTH/2, GHEIGHT/2, Xpos, Ypos)
PLOT( Xpos, Ypos )
REPEAT
END
```



DATA

Purpose

Store constant data.

Syntax

DATA *value* {, *value* }

Description

Stores numerical and string constants for later retrieval using the READ command.

Associated

DATA, READ, RESTORE

Example

```
REM Load the name of the days of the
REM week into a string array
DATA "Monday", "Tuesday", "Wednesday"
DATA "Thursday", "Friday", "Saturday"
DATA "Sunday"
DIM DaysOfWeek$( 7 )
FOR DayNo = 1 TO 7 LOOP
   READ DaysOfWeek$( DayNo )
REPEAT
FOR DayNo = 1 TO 7 LOOP
   PRINT "Day of the week number "; DayNo;
   PRINT " is "; DaysOfWeek$( DayNo )
REPEAT
END
```

DATE$

Purpose

Return the current date.

Syntax

*todaysdate$* = DATE$

Description

This returns a string with the current date in the format: YYYY-MM-DD. For example: 2015-03-24.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
PRINT "Today is "; FN FormatDate(DATE$)
END
DEF FN FormatDate()
   DayNo = VAL( RIGHT$( DATE$, 2 ))
   MonthNo = VAL( MID$( DATE$, 5, 2 ))
   Year$ = LEFT$( DATE$, 4 )
   SWITCH ( DayNo MOD 10 )
      CASE 1
        DaySuffix$ = "st"
      ENDCASE
      CASE 2
        DaySuffix$ = "nd"
      ENDCASE
      CASE 3
        DaySuffix$ = "rd"
      ENDCASE
      DEFAULT
        DaySuffix$ = "th"
      ENDCASE
   ENDSWITCH
   FOR I = 1 TO MonthNo LOOP
      READ MonthName$
   REPEAT
   Result$ = STR$( DayNo ) + DaySuffix$ + " "
= Result$ + MonthName$ + " " + Year$
DATA "January","February","March","April"
DATA "May","June","July","August"
DATA "September","October","November"
DATA "December"
```

DEF FN

Purpose

Create a user defined function.

Syntax

DEF FN *name*({*parameter*}{,*parameter*})
  {*commands*}
=*value*

Description

User defined functions are similar to user defined procedures except that they can return a value. This can be either a number or a character string.

Associated

DEF FN, FN, LOCAL

Example

```
REM Function test - print squares
FOR I = 1 TO 10 LOOP
  x = FN square( I )
PRINT I; " squared is "; x
REPEAT
END
DEF FN square( num )
  LOCAL result
  result = num * num
= result
```

DEF PROC

Purpose

Create a user defined procedure.

Syntax

DEF PROC *name*({ *parameter* } {, *parameter*})
  { *commands* }
ENDPROC

Description

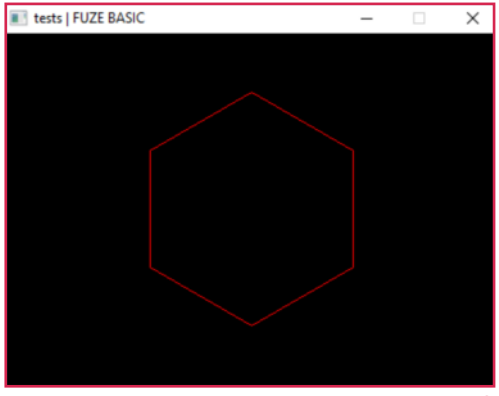
Allows you to create your own routines that can be called by their label. Once you have written a procedure to do a particular task you can copy it into other programs that require it. Procedures are usually defined after the END of the program.

Associated

DEF PROC, LOCAL, PROC, ENDPROC

Example

```
CLS
PROC Hexagon( 200, 200, 100, Red)
UPDATE
END
DEF PROC Hexagon( x, y, l, c )
  PENUP
  MOVETO( x-l * COS( 30 ), y + l/2 )
  COLOUR = c
  PENDOWN
  FOR l = 1 TO 6 LOOP
    RIGHT( 60 )
    MOVE( l )
  REPEAT
ENDPROC
```



DEFCHAR

Purpose

Define a new font character.

Syntax

DEFCHAR(*char*, *line1* ... *line10*)

Description

Create a user defined font character. The *char* parameter is the position of the character within the font (0-255) e.g. 65 is the capital letter A in ASCII. The character consists of 10 lines with 8 pixels in each line. These are set by the corresponding bits in each of the *line* parameters. So for example decimal 170 (binary 10101010) would set alternate pixels on the corresponding line of the character.

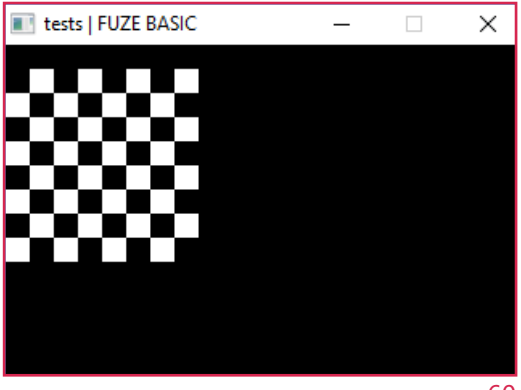
NOTE: Character zero 'CHR$(0)' should not be used.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
REM Define Chessboard Character
FONTSIZE( 10 )
DEFCHAR( 2, 0, 85, 170, 85, 170, 85, 170, 85, 170, 0 )
PRINT CHR$( 2 )
END
```



DEG

Purpose

Set angle units to degrees.

Syntax

DEG

Description

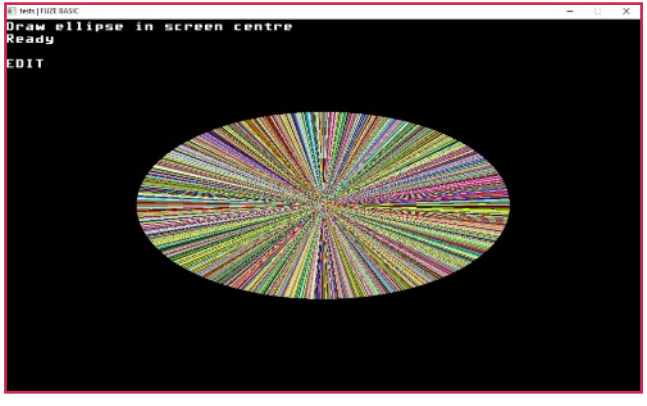
Switches the internal angle system to degree mode. There are 360 degrees in a full circle.

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
CLS
PRINT "Draw ellipse in screen centre"
DEG
FOR Angle = 0 TO 360 STEP 0.01 LOOP
COLOUR=RND(30)
Xpos = 300 * COS( Angle ) + GWIDTH / 2
Ypos = 150 * SIN( Angle ) + GHEIGHT / 2
LINE (GWIDTH/2, GHEIGHT/2, Xpos, Ypos)
PLOT( Xpos, Ypos )
REPEAT
END
```



DETECTDEVICES

Purpose

Check for BBC micro:bit and Arduino / compatible devices.

Syntax

PRINT DETECTDEVICES

Description

Returns value corresponding to the number of compatible devices connected.

Note:

If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

DETECTDEVICES, DEVICETYPE, SETDEVICE

Example

```
devices=DETECTDEVICES
CLS
IF devices = FALSE THEN
PRINT "No devices found"
END
ELSE
PRINT devices; " devices found"
PRINT
FOR ID = 0 TO devices LOOP
IF DEVICETYPE(ID) = 1 THEN
PRINT "BBC micro:bit :ID="; ID
ENDIF
IF DEVICETYPE(ID) = 2 THEN
PRINT "Arduino or compatible device :ID="; ID
ENDIF
REPEAT
END
```

DEVICETYPE

Purpose

Checks which devices are connected (Arduino / compatible or BBC micro:bit)

Syntax

PRINT DEVICETYPE(ID)

Description

Returns a value corresponding to the device type:

1 = BBC micro:bit
2 = Arduino or compatible device

Note:

If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

DETECTDEVICES, DEVICETYPE, SETDEVICE

Example

```
devices=DETECTDEVICES
CLS
IF devices = FALSE THEN
PRINT "No devices found"
END
ELSE
PRINT devices; " devices found"
PRINT
FOR ID = 0 TO devices LOOP
IF DEVICETYPE(ID) = 1 THEN
PRINT "BBC micro:bit :ID="; ID
ENDIF
IF DEVICETYPE(ID) = 2 THEN
PRINT "Arduino or compatible device :ID="; ID
ENDIF
REPEAT
END
```

DIGITALREAD

Purpose

Read the state of a digital pin on the Raspberry Pi.

Syntax

pinvalue = DIGITALREAD(*pinno*)

Description

Reads the state of a digital pin on the Raspberry Pi. You may need to set the pin mode beforehand to make sure it's configured as an input device. It will return TRUE or FALSE to indicate an input being high or low respectively.

Note:

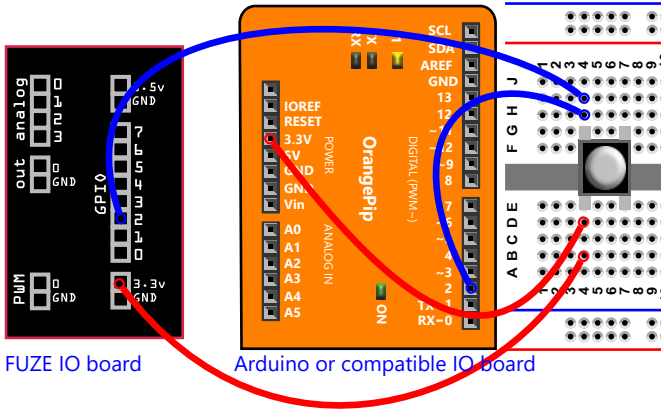
If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

ANALOGREAD, ANALOGWRITE, DIGITALREAD, DIGITALWRITE, PINMODE, PWMWRITE, SOFTPWMWRITE

Example

```
REM Set pin 2 to input
PINMODE( 2, 0 )
REM Wait for button to be pushed
UNTIL DIGITALREAD( 2 ) LOOP
REPEAT
PRINT "Button Pushed"
END
```



FUZE IO board

Arduino or compatible IO board

DIGITALWRITE

Purpose

Set the state of a digital pin on the Raspberry Pi.

Syntax

DIGITALWRITE(*pinno*, *pinvalue*)

Description

This procedure sets a digital pin to the supplied value - 0 for off or 1 for on. As with DIGITALREAD, you may need to set the pin mode (to output) beforehand.

Note:

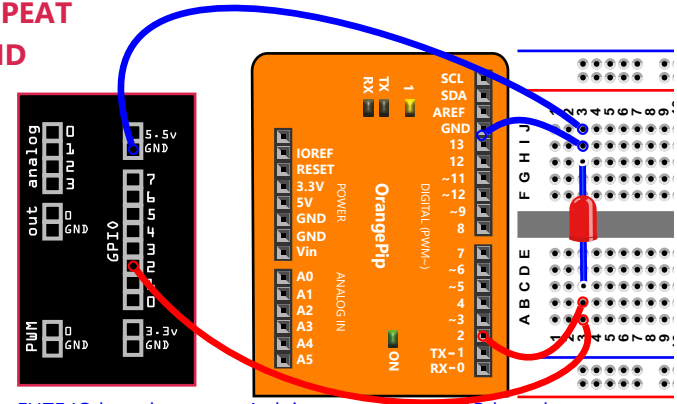
If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

ANALOGREAD, ANALOGWRITE, DIGITALREAD, DIGITALWRITE, PINMODE, PWMWRITE, SOFTPWMWRITE

Example

```
REM Flash LED attached to pin2 of GPIO
REM Set pin 2 to output mode
PINMODE( 2, 1 )
LOOP
  REM Set output High ( on )
  DIGITALWRITE( 2, 1 )
  WAIT( 1 )
  REM Set output Low ( off )
  DIGITALWRITE( 2, 0 )
  WAIT( 1 )
REPEAT
END
```



FUZE IO board

Arduino or compatible IO board

DIM

Purpose

Dimension an array of variables.

Syntax

DIM *variable*(dimension {, dimension })

Description

Creates an indexed variable with one or more dimensions. The variable can be either a numeric or character string type (they cannot hold mixed values). The index is a number from 0 to the size of the dimension. Associative arrays (sometimes called maps) are another way to refer to the individual elements of an array. In the example below we use a number, however strings are also allowed. They can be multi-dimensional and you can freely mix numbers and strings for the array indices.

Example

```
REM Initialise the squares of a chess
REM board to black or white
DIM ChessBoard( 8, 8 )
Count = 0
FOR Row = 1 TO 8 LOOP
  FOR Col = 1 TO 8 LOOP
    Count = Count + 1
    IF Count MOD 2 = 1 THEN
      ChessBoard( Row, Col ) = Black
    ELSE
      ChessBoard( Row, Col ) = White
    ENDIF
  REPEAT
REPEAT
PRINT ChessBoard( 1, 4 )
END
```

ELLIPSE

Purpose

Draw an ellipse on the screen.

Syntax

ELLIPSE(*xpos*, *ypos*, *xradius*, *yradius*, *fill*)

Description

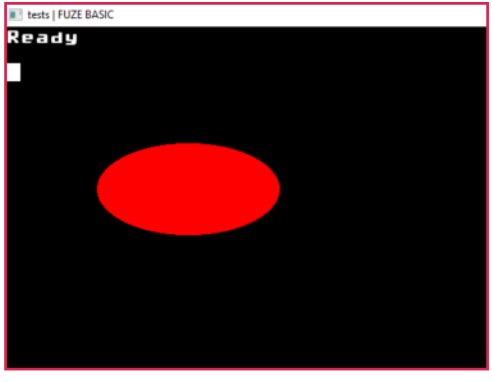
Draws an ellipse centred at position (*xpos*, *ypos*) with the specified *xradius* and *yradius* in the current foreground COLOUR. The final parameter *fill* is either TRUE or FALSE, and specifies filled (TRUE) or outline (FALSE).

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
REM Draw a filled red ellipse at
REM location 200, 200
COLOUR = red
ELLIPSE( 200, 200, 100, 50, TRUE )
UPDATE
END
```



ELSE

Purpose

Execute statement(s) when a tested condition is False.

Syntax

IF *condition* THEN
{ *statements* }
ELSE
{ *statements* }
ENDIF

Example

```
Number = 13
IF Number MOD 2 = 0 THEN
   PRINT "Number is Even"
ELSE
   PRINT "Number is Odd"
ENDIF
END
```

Associated

CASE, DEFAULT, ELSE, ENDIF, ENDCASE, ENDSWITCH, IF THEN, SWITCH

END

Purpose
End program execution.

Syntax
END

Description
Program execution is ended. Programs must terminate with the END or STOP commands or an error will occur.

Associated
STOP, CONTINUE, BREAK

Example

```
PRINT "Hello World"
END
```

ENDIF

Purpose

Terminate a multiline conditional statement.

Syntax

IF *condition* THEN
{ *statements* }
ENDIF

Description

We can extend the IF statement over multiple lines, if required. The way you do this is by making sure there is nothing after the THEN statement and ending it all with the ENDIF statement.

Associated

CASE, DEFAULT, ELSE, ENDIF, ENDCASE, ENDSWITCH, IF THEN, SWITCH

Example

```
DayOfWeek = 5
IF DayOfWeek < 6 THEN
   PRINT "It is a Weekday"
   PRINT "Go to Work!"
ENDIF
END
```

ENDPROC

Purpose

Defines the end of a PROCedure

Syntax

ENDPROC

Description

End a PROCedure and return to the next command after the procedure was called.

Associated

DEF PROC, LOCAL, PROC, ENDPROC

Example

```
CLS
PROC hello
END
DEF PROC hello
PRINTAT ( 10,10 ); "Hello"
ENDPROC
```

EOF

Purpose

Return true if the end of an input file has been reached.

Syntax

endoffile = EOF(*handle*)

Description

The EOF function returns a TRUE or FALSE indication of the state of the file pointer when reading the file. It is an error to try to read past the end of the file, so if you are reading a file with unknown data in it, then you must check at well defined intervals (e.g. Before each INPUT#).

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN( "eoftest.txt" )
FOR r = 0 TO 10 LOOP
   PRINT# handle, "Record "; r
REPEAT
CLOSE ( handle )
handle = OPEN( "eoftest.txt" )
WHILE NOT EOF ( handle ) LOOP
   INPUT# handle, record$
   PRINT record$
REPEAT
CLOSE ( handle )
END
```

EXP

Purpose

Return the exponential value of the specified number.

Syntax

exponential = EXP(*number*)

Description

Returns the exponential value of the specified *number*. This is e to the power of *number* where e is the exponential constant (approximately 2.718281828). The exponential function arises whenever a quantity grows or decays at a rate proportional to its current value. This is the opposite of the LOG function
i.e. EXP(LOG(X)) = X

Associated

LOG

Example

```
REM prints 2.718281828
PRINT EXP( 1 )
REM prints 22026.46579
PRINT EXP( 10 )
REM prints 10
PRINT LOG( EXP( 10 ))
END
```

FADE

Purpose

Fades the screen display

Syntax

FADE (*percent*)

Description

Associated

FADE

Example

FALSE

Purpose

Represent the logical "false" value.

Syntax

FALSE

Description

Represents a Boolean value that fails a conditional test. It is equivalent to a numeric value of 0.

Associated

TRUE

Example

```
condition = FALSE
IF condition = FALSE THEN
   PRINT "Condition is FALSE"
ENDIF
IF NOT condition THEN
   PRINT "Condition is FALSE"
ENDIF
PRINT "Condition= ";condition
END
```

FFWD

Purpose

Move the file pointer to the end of a file.

Syntax

FFWD(*handle*)

Description

Move the file pointer back to the end of the file specified by *handle*. If you want to append data to the end of an existing file, then you need to call FFWD before writing the data.

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN( "ffwdtest.txt" )
PRINT# handle, "First Line"
CLOSE ( handle )
handle = OPEN ("ffwdtest.txt")
FFWD ( handle )
PRINT# handle, "Appended line"
CLOSE ( handle )
handle = OPEN( "ffwdtest.txt" )
WHILE NOT EOF ( handle ) LOOP
   INPUT# handle, record$
   PRINT record$
REPEAT
CLOSE ( handle )
END
```

FN

Purpose

Call a user defined function.

Syntax

result=FN *name*({ *argument* } {, *argument* })

Description

Calls the specified user defined function called *name* with the specified *arguments*. The returned *result* can then be used by the program. Once the function has been executed control returns to the command following.

Associated

DEF FN, FN, LOCAL

Example

```
PRINT FN SphereVolume( 10 )
END
REM Function calculate volume of a sphere
REM with radius r
DEF FN SphereVolume( r )
= ( 4 / 3 ) * PI * r * r * r
```

FONTSIZE

Purpose

Scale the text font.

Syntax

FONTSIZE(*scale*)

Description

Change the size of the text font. Maximum 20.

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

Example

```
CLS
FOR size = 1 TO 7 LOOP
   INK = RND(30)
   FONTSIZE( size )
   PRINT "Hello World"
REPEAT
END
```

FOR LOOP

Purpose

Loop a specified number of times using a counter.

Syntax

FOR *count* = *start* TO *end* [STEP *step*] LOOP
  statements
REPEAT

Description

The *count* variable is initially set to *start* and changes by *step* each time around the loop until *count* is greater than or equal to *end*. The optional *STEP*, which defaults to 1 may be less than zero to count backwards. The end of the loop is indicated using the REPEAT.

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT UNTIL, UNTIL REPEAT, WHILE REPEAT

Example

```
REM year into a string array
DATA "January", "February", "March"
DATA "April", "May", "June"
DATA "July", "August", "September"
DATA "October", "November", "December"
DIM Months$( 12 )
FOR Month = 1 TO 12 LOOP
  READ Months$( Month )
REPEAT
PRINT "The seventh month is "; Months$( 7 )
END
```

FREEIMAGE

Purpose

Release an image from memory

Syntax

FREEIMAGE(*handle*)

Description

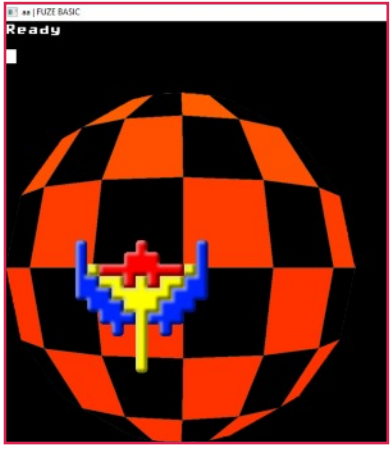
Frees up the memory space taken up by a stored image

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
CLS
handle = LOADIMAGE( "ball.png" )
PLOTIMAGE( handle, 0, 0 )
UPDATE
FREEIMAGE( handle )
handle = LOADIMAGE( "fighter.png" )
PLOTIMAGE( handle, 100, 100 )
UPDATE
END
```



FULLSCREEN

Purpose

Sets the display to full screen mode.

Syntax

FULLSCREEN = { TRUE / FALSE }

Description

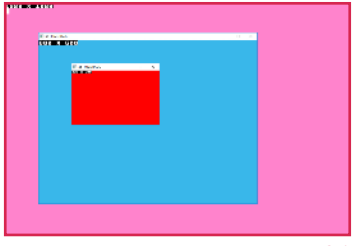
Switches between full screen or windowed mode. Note this does not set the resolution to the screen display mode so unless you set the mode manually you will get a border.

Associated

FULLSCREEN, GHEIGHT, GWIDTH, ORIGIN, SETMODE

Example

```
SETMODE( 320, 200 )
COLOUR = RED
RECT( 0, 0, GWIDTH, GHEIGHT, 1 )
PRINT GWIDTH; " X "; GHEIGHT
UPDATE
WAIT( 2 )
SETMODE( 800, 600 )
COLOUR = AQUA
RECT( 0, 0, GWIDTH, GHEIGHT, 1 )
PRINT GWIDTH; " X "; GHEIGHT
UPDATE
WAIT( 2 )
FULLSCREEN = 1
COLOUR = PINK
RECT( 0, 0, GWIDTH, GHEIGHT, 1 )
PRINT GWIDTH; " X "; GHEIGHT
UPDATE
```



GET

Purpose

Get a single character code from the keyboard.

Syntax

asciicode = GET

Description

This pauses program execution and waits for you to type a single character on the keyboard, then returns the value of the key pressed as a numeric variable (ASCII).

Associated

CLEARKEYBOARD, GET, GET$, INKEY, INPUT, SCANKEYBOARD

Example

```
PRINT "Press a key"
key = GET
PRINT "ASCII Value of key = "; key
END
```

GET$

Purpose

Get a single character from the keyboard.

Syntax

*key$* = GET$

Description

This pauses program execution and waits for you to type a single character on the keyboard, then returns the key as a string variable.

Example

```
PRINT "Press a key"
key$ = GET$
PRINT "You Pressed Key: "; key$
END
```

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

GETDISTANCE

Purpose

Find the distance between two points.

Syntax

value = GETDISTANCE(x1, y1, x2, y2)

Description

Returns the distance between two specified screen locations.

Associated

GETDISTANCE, GETSPRITEDISTANCE

Example

```
UPDATEMODE = 0
LOOP
 CLS2
 x = MOUSEX
 y = MOUSEY
 xx = GWIDTH / 2
 yy = GHEIGHT / 2
 COLOUR = GREY
 CIRCLE (xx, yy, 50, 1)
 COLOUR = WHITE
 LINE (xx, yy, x, y)
 COLOUR = RED
 CIRCLE (x, y, 50, 1)
 dist = GETDISTANCE (x, y, xx, yy)
 PRINTAT (0, 0);  "Distance=";  dist
 UPDATE
REPEAT
```

GETSPRITEDISTANCE

Purpose

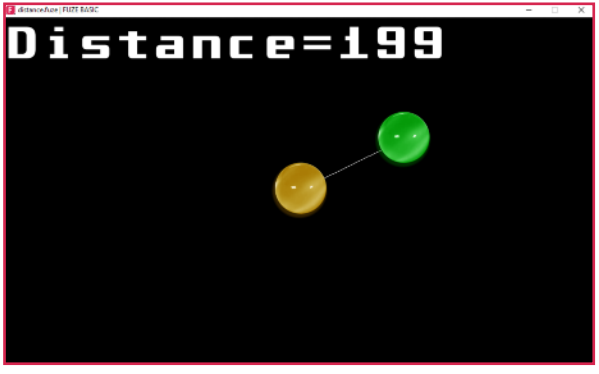
Find the distance between two sprites points.

Syntax

value = GETSPRITEDISTANCE(sprite1, sprite2)

Description

Returns the distance between two specified sprite positions.



Associated

GETDISTANCE, GETSPRITEDISTANCE

Example

```
UPDATEMODE = 0
sp1=NEWSPRITE( 1 )
LOADSPRITE("gem0.png", sp1, 0)
SETSPRITEORIGIN(sp1,GETSPRITEW(sp1)/2,GETSPRITEH(sp1)/2)
sp2=NEWSPRITE(1)
LOADSPRITE("gem4.png", sp2, 0)
SETSPRITEORIGIN(sp2,GETSPRITEW(sp2)/2,GETSPRITEH(sp2)/2)
LOOP
 CLS2
 x = MOUSEX
 y = MOUSEY
 xx = GWIDTH / 2
 yy = GHEIGHT / 2
 PLOTSPRITE (sp1, xx, yy, 0)
 COLOUR = WHITE
 LINE ( xx,  yy, x, y)
 PLOTSPRITE (sp2, x, y, 0)
 dist = INT(GETSPRITEDISTANCE (x, y, xx, yy))
 PRINTAT (0, 0);  "Distance=";  dist
 UPDATE
REPEAT
```

GETIMAGEH / GETIMAGEW

Purpose

Get the pixel height and width of a loaded image.

Syntax

Value1 = GETIMAGEH(*handle*)
Value2 = GETIMAGEW(*handle*)

Description

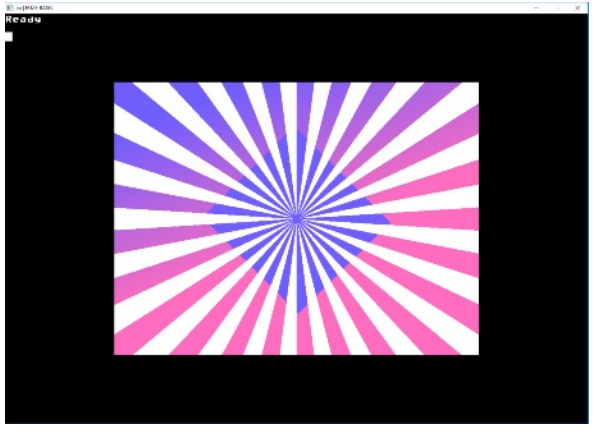
Gets the height and or width in pixels of a loaded image
.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
REM Centre image on screen
SETMODE(1280,900)
logo = LOADIMAGE( "img.png" )
imageW = GETIMAGEW( logo )
imageH = GETIMAGEH( logo )
X = ( GWIDTH - imageW ) / 2
Y = ( GHEIGHT - imageH ) / 2
PLOTIMAGE( logo, X, Y )
UPDATE
END
```



GETMOUSE

Purpose

Read values from an attached mouse

Syntax

GETMOUSE(*xpos*, *ypos*, *buttons*)

Description

Read the current state of the mouse. *xpos* is the horizontal mouse position, *ypos* is the vertical position and *buttons* is the state of the mouse buttons.

The button values are;

0 - no buttons pressed
1 - left button pressed
2 - middle button pressed
3 - left & middle buttons pressed
4 - right button pressed
5 - left & right buttons pressed
6 - right & middle button pressed
7 - left, middle & right buttons pressed

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
CLS
LOOP
  GETMOUSE (x, y, b)
  CIRCLE (x, y, 30, 1)
 REM LOOP colour if left button pressed
  IF b = 1 THEN COLOUR = COLOUR MOD 30 + 1
 REM Exit if right button pressed
UPDATE
REPEAT UNTIL b = 4
END
```

GETPIXEL

Purpose

Return the colour of the specified pixel.

Syntax

colour = GETPIXEL(*xpos*, *ypos*)

Description

This returns the internal colour code (0 - 15) of the pixel at the specified point (*xpos*, *ypos*). This is for the "named" colours - e.g. Red, Green etc. It returns -1 if the pixel colour is not a standard colour - in which case, you need to use GETPIXELRGB.

Associated

GETPIXELRGB

Example

```
CLS
COLOUR = RED
Xpos = GWIDTH / 2
Ypos = GHEIGHT / 2
PLOT( Xpos, Ypos )
PRINT GETPIXEL( Xpos, Ypos )
END
```

GETPIXELRGB

Purpose

Return the RGB colour of the specified pixel.

Syntax

RGBcolour = GETPIXELRGB(*xpos*, *ypos*)

Description

This returns the RGB colour of the pixel at the specified point (*xpos*,*ypos*). This will return a 24-bit value.

Associated

GETPIXEL

Example

```
CLS
RGBCOLOUR( 49, 101, 206 )
Xpos = GWIDTH / 2
Ypos = GHEIGHT / 2
CIRCLE( Xpos, Ypos, 50, TRUE )
pixel = GETPIXELRGB( Xpos, Ypos )
PRINT "RGB = ";    pixel
PRINT "Red = ";    ( pixel >> 16 ) & 0xFF
PRINT "Green = "; ( pixel >> 8 ) & 0xFF
PRINT "Blue = "; ( pixel >> 0 ) & 0xFF
END
```

GETSPRITEANGLE

Purpose

Returns a sprite's current angle

Syntax

GETSPRITEANGLE(*spriteIndex*)

Description

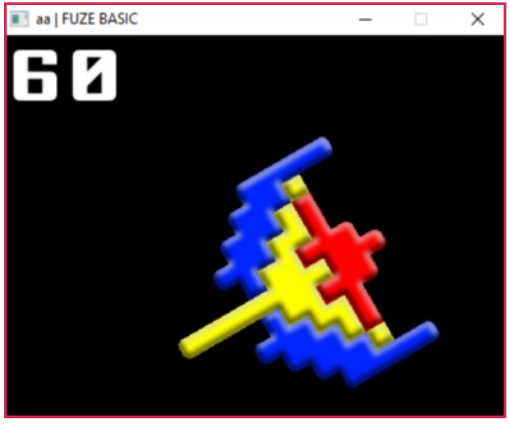
Returns the angle of the sprite with the specified *spriteIndex*.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
sprite = NEWSPRITE ( 1 )
LOADSPRITE ( "fighter.png", sprite, 0 )
angle = 0
PLOTSPRITE ( sprite, GWIDTH / 2, GHEIGHT / 2, 0 )
LOOP
CLS2
  SETSPRITEANGLE ( sprite, angle )
  angle = angle + 1
  IF angle > 359 THEN angle = 0
  PRINTAT( 0,0 ); GETSPRITEANGLE( sprite )
  UPDATE
REPEAT
```



GETSPRITEH / GETSPRITEW

Purpose

Get the pixel height and width of a sprite.

Syntax

value1 = GETSPRITEH(*spriteIndex*)

value2 = GETSPRITEW(*spriteIndex*)

Description

Returns the height and or width in pixels of the sprite with the specified *spriteIndex*.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
REM Centre sprite on the screen
CLS
index = NEWSPRITE( 1 )
fuzelogo$ = "logo.png"
LOADSPRITE( fuzelogo$, index, 0 )
spriteW = GETSPRITEW( index )
spriteH = GETSPRITEH( index )
X=( GWIDTH - spriteW) / 2
Y=( GHEIGHT - spriteH ) / 2
PLOTSPRITE( index, X, Y, 0 )
UPDATE
END
```

GETSPRITEPIXELRGBA

Purpose

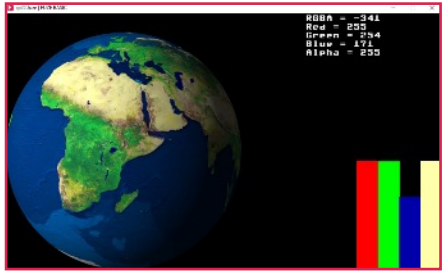
Get a single pixel colour value from a specified sprite.

Syntax

value = GETSPRITEPIXELRGBA(*spriteID, subSpriteID, x, y*)

Description

Returns a 32BIT Red, Green, Blue & Alpha value from the specified x, y position, relative to the sprite's lower left coordinate.



Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
UPDATEMODE = 0
earth = NEWSPRITE (1)
LOADSPRITE ("planetEarth.png", earth, 0)
LOOP
 CLS2
 PLOTSPRITE (earth, 0, 0, 0)
 IF LEFTCLICK THEN SPRITECENTRE (earth)
 IF RIGHTCLICK THEN SETSPRITEORIGIN (earth, 0, 0)
 px = GETSPRITEPIXELRGBA (earth, 0, MOUSEX, MOUSEY)
 alpha = (px >> 24) & 0xFF
 rp = (px >> 16) & 0xFF
 gp = (px >> 8) & 0xFF
 bp = (px >> 0) & 0xFF
 PRINTAT (TWIDTH - 20, 0);  "RGBA = ";  px
 PRINTAT (TWIDTH - 20, 1);  "Red = ";  rp
 PRINTAT (TWIDTH - 20, 2);  "Green = ";  gp
 PRINTAT (TWIDTH - 20, 3);  "Blue = ";  bp
 PRINTAT (TWIDTH - 20, 4);  "Alpha = ";  alpha
 RGBCOLOUR (rp, 0, 0)
 RECT (GWIDTH - 200, 0, 50, rp, 1)
 RGBCOLOUR (0, gp, 0)
 RECT (GWIDTH - 150, 0, 50, gp, 1)
 RGBCOLOUR (0, 0, bp)
 RECT (GWIDTH - 100, 0, 50, bp, 1)
 RGBCOLOUR (rp, gp, bp)
 RECT (GWIDTH - 50, 0, 50, 255, 1)
 UPDATE
REPEAT
```

GETSPRITEX / GETSPRITEY

Purpose

Get the X and Y position of a sprite.

Syntax

value1 = GETSPRITEX(*spriteIndex*)
value2 = GETSPRITEY(*spriteIndex*)

Description

Returns the X and or Y position in pixels of the sprite with the specified *spriteIndex*.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
UPDATEMODE = 0
sprite = NEWSPRITE ( 1 )
LOADSPRITE ( "fighter.png", sprite, 0 )
LOOP
    CLS2
    PLOTSPRITE ( sprite, GWIDTH / 2, GHEIGHT / 2, 0 )
    PRINT GETSPRITEX ( sprite )
    PRINT GETSPRITEY ( sprite )
    UPDATE
REPEAT
```

GHEIGHT / GWIDTH

Purpose

Find the current height and width of the display.

Syntax

height = GHEIGHT
Width = GWIDTH

Description

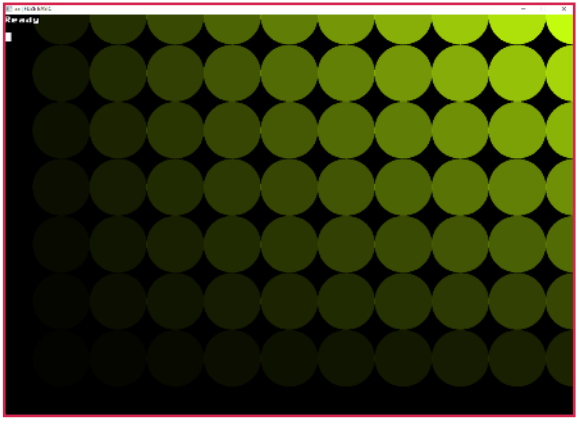
These used to find the current height and or width of the display in either high resolution or low resolution pixels.

Associated

FULLSCREEN, GHEIGHT, GWIDTH, ORIGIN, SETMODE

Example

```
REM Draw lots of alpha blended circles.
CLS
radius=GWIDTH/20
alpha=256/(GWIDTH*GHEIGHT)
FOR y = 0 TO GHEIGHT STEP radius LOOP
FOR x= 0 TO GWIDTH STEP radius LOOP
COLOUR = LIME
SETALPHA(alpha*(x*y))
CIRCLE( x, y, radius/2, TRUE )
UPDATE
REPEAT
REPEAT
END
```



GRABREGION

Purpose

Grab a region of the screen to a temporary buffer

Syntax

handle = GRABREGION(*x*, *y*, *width*, *height*)

Description

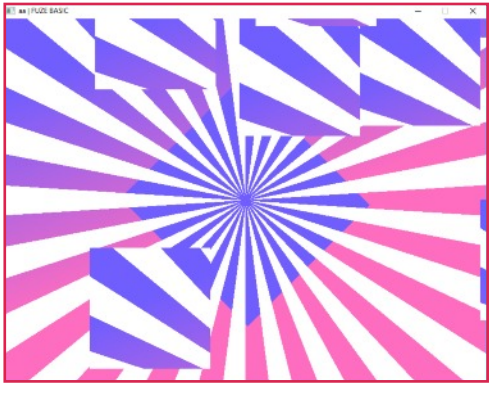
Grab a region of the screen with x and y as the location and with *width* and *height* in pixels. The region can be recalled by its handle and pasted using PLOTIMAGE.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
image=LOADIMAGE("img.png")
SETMODE(GETIMAGEW(image), GETIMAGEH(image))
PLOTIMAGE(image, 0, 0)
UPDATE
box = GRABREGION( 0, 0, 200, 200 )
LOOP
PLOTIMAGE( box, RND(GWIDTH), RND(GHEIGHT))
UPDATE
WAIT(0.1)
REPEAT
END
```



HIDESPRITE

Purpose

Remove a sprite from the screen.

Syntax

HIDESPRITE(*spriteindex*)

Description

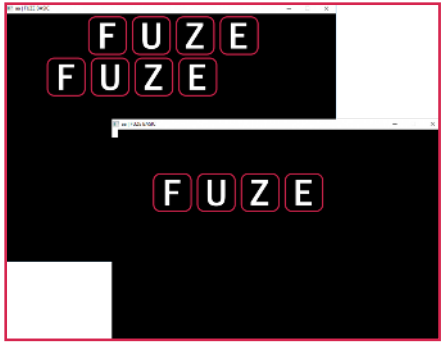
This removes the sprite at the specified *spriteindex* from the screen. You do not have to erase a sprite from the screen when you move it, just call PLOTSPRITE with the new coordinates.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
fuzelogo$ = "logo.png"
sprite1 = NEWSPRITE ( 1 )
sprite2 = NEWSPRITE ( 1 )
LOADSPRITE ( fuzelogo$, sprite1, 0 )
LOADSPRITE ( fuzelogo$, sprite2,0 )
PLOTSPRITE ( sprite1, 100, 100, 0 )
PLOTSPRITE ( sprite2, 200, 200, 0 )
UPDATE
WAIT( 2 )
REM Remove a sprite from the screen
HIDESPRITE ( sprite2 )
UPDATE
END
```



HLINE

Purpose

Draw a horizontal line.

Syntax

HLINE(*xpos1*, *xpos2*, *ypos*)

Description

Draws a horizontal line on row y, from column xpos1 to column xpos2.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
COLOUR = RED
FOR ypos = 0 TO GHEIGHT STEP 100 LOOP
   HLINE( 0, GWIDTH, ypos )
REPEAT
UPDATE
END
```

HTAB / VTAB

Purpose

Set/Read the current text cursor horizontal and vertical positions.

Syntax

HTAB = *value1*

value1 = HTAB

VTAB = *value2*

value2 = VTAB

Description

Set/Read the current text cursor horizontal and vertical positions.

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

Example1

```
CLS
FOR xpos = 0 TO TWIDTH STEP 2 LOOP
   HTAB = xpos
   PRINT HTAB
REPEAT
END
```

Example2

```
CLS
FOR ypos = 0 TO THEIGHT LOOP
   VTAB = ypos
   PRINT VTAB
REPEAT
END
```

HVTAB

Purpose

Move the current text cursor to the specified position.

Syntax

HVTAB(*xpos*, *ypos*)

Description

The cursor is moved to the supplied column *xpos* and line *ypos*. Note that (0,0) is top-left of the text screen.

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

Example

```
CLS
FONTSIZE(5)
INK = PINK
HVTAB( TWIDTH / 2 - LEN("CENTRE")/2, THEIGHT / 2 )
PRINT "CENTRE"
HVTAB( 0, 0 )
END
```

IF THEN

Purpose

Execute a statement conditionally.

Syntax

IF *condition* THEN {*statement*}

Description

The statement is executed when the condition evaluates to TRUE (not 0). Unlike some implementations of BASIC the THEN is required.

Example

```
PRINT "Press Space Bar to Continue"
LOOP
    IF INKEY = 32 THEN BREAK
REPEAT
PRINT "Space Bar Pressed"
END
```

Associated

CASE, DEFAULT, ELSE, ENDIF, ENDCASE, ENDSWITCH, IF THEN, SWITCH

INK

Purpose

Set/Read the current text foreground colour.

Syntax

foregroundcolour = INK
INK = foregroundcolour

Description

Set/Read the current text foreground (ink) colour.

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

Example

```
CLS
string$ = "This is multicoloured text"
FOR I = 1 TO LEN( string$ ) LOOP
   INK = I MOD 30 + 1
   PRINT MID$( string$, I - 1 , 1 );
REPEAT
INK = WHITE
PRINT
END
```

INKEY

Purpose

Get a single character code from the keyboard without pausing.

Syntax

asciicode = INKEY

Description

This is similar to GET except that program execution is not paused; If no key is pressed, then -1 is returned. The following constants are predefined to test for special keys: KeyUp, KeyDown, KeyLeft, KeyRight, KeyIns, KeyHome, KeyDel, KeyEnd, KeyPgUp, KeyPgDn, KeyF1, KeyF2, KeyF3, KeyF4, KeyF5, KeyF6, KeyF7, KeyF8, KeyF9, KeyF10, KeyF11, KeyF12.

Associated

CLEARKEYBOARD, GET, GET$, INKEY, INPUT, SCANKEYBOARD

Example

```
REM Show the ASCII code for the last key pressed
LastKey = -1
REM Press Esc to Quit
LOOP
  Key=INKEY
  IF Key<> - 1 AND Key <> LastKey THEN
    PRINT "Key Pressed: "; Key
    LastKey = Key
  ENDIF
REPEAT
```

INPUT

Purpose

Read data from the keyboard into a variable.

Syntax

INPUT [*prompt$*,] *variable*

Description

When FUZE BASIC encounters the INPUT statement, program execution stops, a question mark(?) is printed and it waits for you to type something. It then assigns what you typed to the variable. If you typed in a string when it was expecting a number, then it will assign zero to the number. To stop it printing the question mark, you can optionally give it a string to print.

Associated

CLEARKEYBOARD, GET, GET$, INKEY, INPUT, SCANKEYBOARD

Example

```
INPUT "What is your Name? ", Name$
PRINT "Hello " + Name$
END
```

INPUT#

Purpose

Read data from a file.

Syntax

INPUT# *handle*, *variable*

Description

This works similarly to the regular INPUT instruction, but reads from the file identified by the supplied *handle* rather than from the keyboard. Note that unlike the regular keyboard INPUT instruction, INPUT# can only read one variable at a time.

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN( "testfile.txt")
PRINT# handle, "Hello World"
REWIND( handle )
INPUT# handle, record$
PRINT record$
CLOSE ( handle )
END
```

INT

Purpose

Return the integer part of a number.

Syntax

integerpart = INT(*number*)

Description

Returns the integer part of the specified *number*.

Example

```
PRINT "The integer part of PI is ";
PRINT INT( PI )
END
```

LEFT

Purpose

Turns the turtle to the left (counter clockwise) by the given angle.

Syntax

LEFT(*angle*)

Description

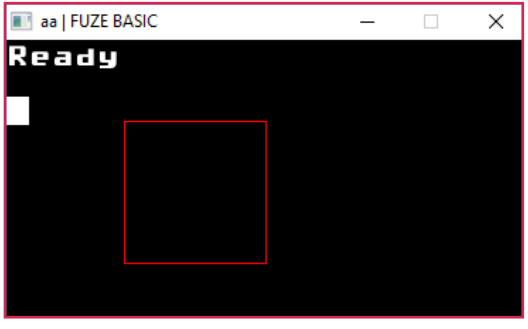
Turns the virtual graphics turtle to the left (counter clockwise) by the given *angle* in the current angle units.

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE

Example

```
REM Draw a box using turtle graphics
CLS
COLOUR = RED
MOVE( 50 )
DEG
LEFT( 90 )
MOVE( 50 )
PENDOWN
FOR I = 1 TO 4 LOOP
   LEFT( 90 )
   MOVE( 100 )
REPEAT
UPDATE
END
```



LEFT$

Purpose

Return the specified leftmost number of a characters from a string.

Syntax

*substring$* = LEFT$(*string$*, *number*)

Description

Returns a sub-string of *string$* with *number* characters from the left (start) of the string. If number is greater than or equal to the length of *string$* then the whole string is returned.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
string$ = "The quick brown fox"
FOR I = 1 TO 20 LOOP
   PRINT LEFT$( string$, I )
REPEAT
END
```

LEFTCLICK

Purpose

Returns TRUE if left mouse button is clicked

Syntax

value = LEFTCLICK

Description

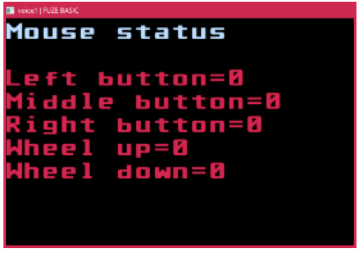
Returns a value of 0 or 1 (TRUE or FALSE) depending on whether it is press or not.

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
UPDATEMODE = 0
FONTSIZE (4)
LOOP
 CLS2
 INK = LIGHTBLUE
 PRINTAT (0, 0);  "Mouse status"
 INK = RASPBERRY
 PRINTAT (0, 2);  "Left button=";  LEFTCLICK
 PRINTAT (0, 3);  "Middle button=";  MIDDLECLICK
 PRINTAT (0, 4);  "Right button=";  RIGHTCLICK
 PRINTAT (0, 5);  "Wheel up=";  WHEELUP
 PRINTAT (0, 6);  "Wheel down=";  WHEELDOWN
 UPDATE
REPEAT
```



LEN

Purpose

Return the length of the specified character string.

Syntax

length=LEN(*string$*)

Description

Returns the number of characters in the specified *string$*.

Example

```
String$="The Quick Brown Fox"
Chars = LEN( String$ )
PRINT "String Length is: "; Chars
FOR I = 0 TO Chars - 1 LOOP
   Char$ = MID$( String$, I, 1 )
   PRINT "Character No. "; I; " is " + Char$
REPEAT
END
```

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

LERP

Purpose

Return an interpolated value between two values.

Syntax

LERP(value1, value2, position)

Description

Returns a value between two numbers determined by the position (between 0 and 1).

Example;

```
PRINT LERP (0, 100, 0.5) //results in 50
PRINT LERP (8, 16, 0.5) //results in 12
PRINT LERP (0, GWIDTH, 0.1) // one tenth of the display width
PRINT LERP (0, 255, 0.5) // If used as a gradient then this is 50% along
```

Associated

SPLINE

Example

```
UPDATEMODE=0
MOUSEON
PAPER=DARKGREY
position=0
LOOP
CLS2
COLOUR=ORANGE
CIRCLE(GWIDTH/2,GHEIGHT/2,10,1)
COLOUR=LIGHTBLUE
CIRCLE(MOUSEX,MOUSEY,20,1)
COLOUR=LIGHTGREEN
cx = LERP(GWIDTH/2,MOUSEX,position)
cy = LERP(GHEIGHT/2,MOUSEY,position)
CIRCLE( cx, cy, 20, 1)
position=position+0.001
IF position>1 THEN position=0
UPDATE
REPEAT
END
```

LINE

Purpose

Draw a line between two points

Syntax

LINE(*xpos1*, *ypos1*, *xpos2*, *ypos2*)

Description

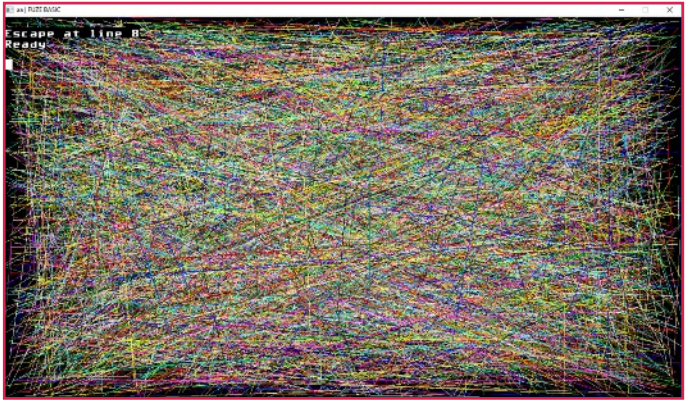
Draw a line between point (*xpos1*, *ypos1*) and point (*xpos2*, *ypos2*) in the current COLOUR.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
LOOP
COLOUR = RND(30)
startX=RND(GWIDTH)
startY=RND(GHEIGHT)
endX=RND(GWIDTH)
endY=RND(GHEIGHT)
LINE( startX, startY, endX, endY)
UPDATE
REPEAT
END
```



LINETO

Purpose

Draw a line from the last point plotted.

Syntax

LINETO(*xpos1*, *ypos1*)

Description

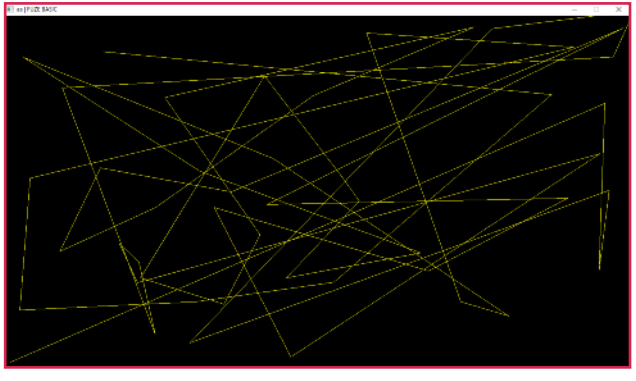
Draws a line from the last point plotted (by the PLOT or LINE procedures) to point (*xpos1*, *ypos1*).

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
COLOUR = YELLOW
PLOT (0, 0)
LOOP
LINETO(RND(GWIDTH), RND(GHEIGHT))
UPDATE
WAIT(0.1)
REPEAT
END
```



LISTDIR

Purpose

Saves a TXT file containing the current folder's file contents.

Syntax

LISTDIR

Description

Saves a TXT file called "files.list" into the current working directory containing file structure within.

Example

```
CLS
LISTDIR
files=OPEN( "files.list" )
WHILE NOT EOF( files ) LOOP
INPUT# files, text$
PRINT text$
REPEAT
CLOSE ( files )
END
```

Associated

LOADFONT

Purpose

Draw a line from the last point plotted.

Syntax

LINETO(*xpos1*, *ypos1*)

Description

Draws a line from the last point plotted (by the PLOT or LINE procedures) to point (*xpos1*, *ypos1*).

NOTE

The fonts used in the example overleaf are stored in the extras folder. We recommend you copy which ever you need to a separate work folder.

Associated

FONTSIZE, LOADFONT

Example


```
FONTSIZE (6)
UPDATEMODE = 0
CLS
LOOP
 CLS2
 INK = RND (30) + 1
 LOADFONT ("A8BitMed.ttf")
 PLOTTEXT  (RND (200), 100, "THIS font IS nice")
 INK = RND (30) + 1
 LOADFONT  ("szxp.ttf")
 PLOTTEXT  (RND (200), 200, "THIS font IS nice")
 INK = RND (30) + 1
 LOADFONT  ("mode7.ttf")
 PLOTTEXT  (RND (200), 350, "THIS font IS nice")
 INK = RND (30) + 1
 LOADFONT ("ZX8.ttf")
 PLOTTEXT (RND (200), 500, "THIS font IS nice")
 WAIT (0.05)
 UPDATE
REPEAT
```

LOADIMAGE

Purpose

Load an image file into memory.

Syntax

handle = LOADIMAGE(*filename*)

Description

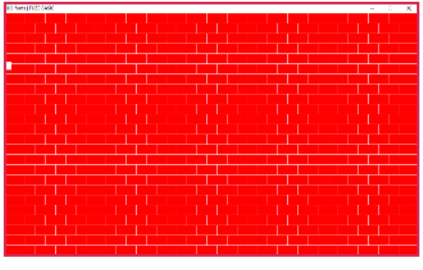
Load an image from a file with the specified *filename*. The returned *handle* can then be used to plot it on the screen with PLOTIMAGE.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
COLOUR = RED
RECT( 0, 0, 50, 50, TRUE )
COLOUR = WHITE
LINE( 0, 50, 50, 50 )
LINE( 0, 25, 50, 25 )
LINE( 0, 25, 0, 50 )
LINE( 50, 25, 50, 50 )
LINE( 25, 0, 25, 25 )
LINE( 0, 0, 50, 0 )
SAVEREGION( "bricks.bmp", 0, 0, 50, 50 )
handle = LOADIMAGE( "bricks.bmp" )
FOR X = 0 TO GWIDTH STEP 50 LOOP
   FOR Y = 0 TO GHEIGHT STEP 50 LOOP
      PLOTIMAGE( handle, X, Y )
   REPEAT
REPEAT
UPDATE
END
```



LOADMUSIC

Purpose

Load a music file into memory ready to be played.

Syntax

handle = LOADMUSIC(*filename*)

Description

Loads an uncompressed music file in wav format (file extension .wav) from the file *filename* into memory and returns a *handle* which can then be used to play the music using the PLAYMUSIC function.

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
handle = LOADMUSIC( "neon.mp3" )
SETMUSICVOL( 70 )
PLAYMUSIC( handle, 1 )
END
```

LOADSAMPLE

Purpose

Load a sound sample into memory ready to be played.

Syntax

handle = LOADSAMPLE(*filename*)

Description

The LOADSAMPLE function loads a sound sample from the uncompressed WAV format file called *filename* and returns a *handle* to it so that it can be played using the PLAYSAMPLE function. You can load up to 32 sound samples into memory at the same time.

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
channel = 0
volume = 70
SETCHANVOL( channel, volume )
intro = LOADSAMPLE( "powerup.wav" )
PLAYSAMPLE( intro, channel, 0 )
WAIT( 4.5 )
END
```

LOADSPRITE

Purpose

Load a sprite from a file into memory.

Syntax

LOADSPRITE(*filename$*, *index*, *subindex*)

Description

This loads a sprite from the supplied *filename$* into memory and associates it with the given sprite *index* and *subindex*. The *index* is the handle returned by a call to NEWSPRITE and the *subindex* is the version of the sprite to allow for animation. The first *subindex* is 0.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
REM Create a new sprite with 1 version
SpriteIndex = NEWSPRITE(1)
REM Load a sprite from a file
Fuzelogo$ = "logo.png"
LOADSPRITE( Fuzelogo$, SpriteIndex, 0 )
REM Draw the sprite on the screen
PLOTSPRITE( SpriteIndex, 200, 200, 0 )
UPDATE
END
```

LOCAL

Purpose

Define variables to be local to a user defined procedure or
function.

Syntax

LOCAL *variable*

Description

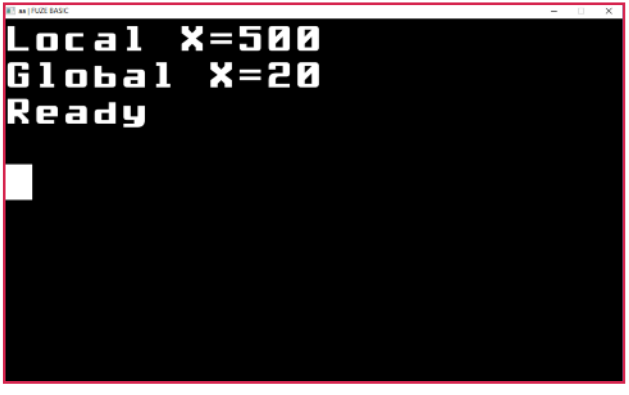
Allows a variable name to be reused in a procedure or function without affecting its value in the calling program.

Associated

DEF FN, DEF PROC, FN, LOCAL

Example

```
CLS
X=20
PROC Test( X )
PRINT "Global X=" + STR$( X )
END
DEF PROC Test( X )
   LOCAL X
   X=500
   PRINT "Local X=" + STR$( X )
ENDPROC
```



LOG

Purpose

Return the natural logarithm of the specified number.

Syntax

naturallogarithm = LOG(*number*)

Description

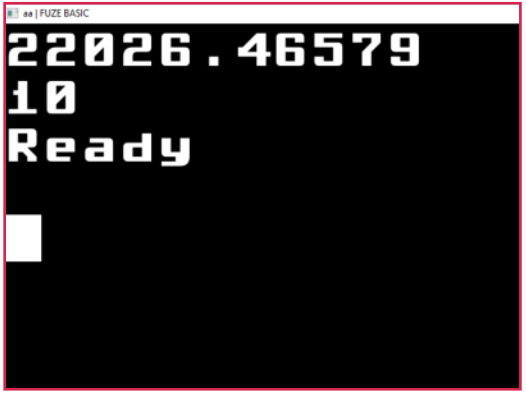
Returns the natural logarithm of the specified *number*.This is the opposite of the EXP function i.e. LOG(EXP(X)) = X. Logarithms are used in science to solve exponential radioactive decay problems and in finance to solve problems involving compound interest.

Associated

EXP

Example

```
X = EXP( 10 )
PRINT X // Will print 22026.46579
PRINT LOG( X ) // Will print 10
END
```



LOCKMOUSE

Purpose

Constrain the mouse point to the display window.

Syntax

LOCKMOUSE([TRUE or FALSE])

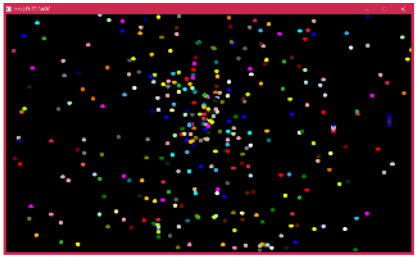
Description

Very useful when using the mouse to control a program as it keeps the mouse focused on the display window.

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example



```
LOCKMOUSE(true)
distance = 1
maxcirc = 360
spacer = 1
size=5
DIM circ1(maxcirc, 3)
FOR n = 0 TO maxcirc STEP spacer LOOP
  circ1(n, 2) = RND (30)
  circ1(n, 3) = n * distance
REPEAT
LOOP
  CLS2
  FOR n = 0 TO maxcirc STEP spacer LOOP
    COLOUR = circ1(n, 2)
    circ1(n, 0) = MOUSEX + circ1(n, 3) * COS (n)
    circ1(n, 1) = MOUSEY + circ1(n, 3) * SIN (n)
    circ1(n, 3) = circ1(n, 3) + distance
    x=ABS (gWidth / 2 - circ1(n, 0))
    y=ABS (gHeight / 2 - circ1(n, 1))
    IF x > gWidth / 2 OR y > gHeight / 2 THEN circ1(n, 3) = 0
    CIRCLE (circ1(n, 0), circ1(n, 1), size, 1)
  REPEAT
  UPDATE
REPEAT
END
```

LOOP REPEAT

Purpose

Create an infinite loop.

Syntax

LOOP
  {*statements*}
REPEAT

Description

Execute the *statements* again and again forever. The BREAK command can be used to terminate a loop. Pressing the Esc key will also interrupt the loop (and program).

Example

```
REM Loop until the space bar is pressed
PRINT "Press the space bar to continue"
LOOP
  IF INKEY = 32 THEN BREAK
REPEAT
END
```

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT UNTIL, UNTIL REPEAT, WHILE REPEAT

MAX

Purpose

Returns the larger of two numbers.

Syntax

maxvalue = MAX(*number1* ,*number2*)

Description

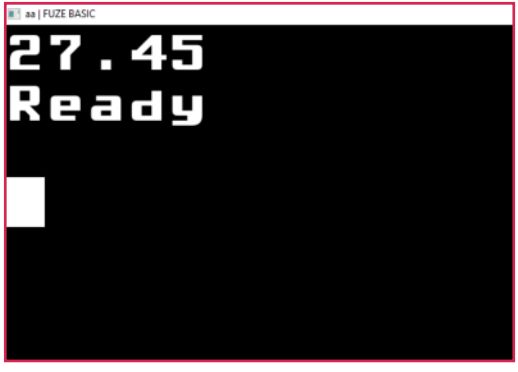
Returns the larger (highest value) of *number1* or *number2*.

Associated

MIN

Example

```
REM Prints the value of number2
number1 = 12.26
number2 = 27.45
PRINT MAX( number1, number2 )
END
```



MICROTIME

Purpose

Find out how long the program has been running in 1,000,000s of a second.

Syntax

value = MICROTIME

Description

This returns a number which represents the time that your program has been running in millionths of seconds.

Associated

TIME

Example

```
REM Simple reaction timer
CLS
WAIT( RND( 3 ) + 1 )
REM Make sure no key pressed
WHILE INKEY <> - 1 LOOP
REPEAT
startTime = MICROTIME
PRINT "Go!"
WHILE INKEY = -1 LOOP
REPEAT
endTime = MICROTIME
PRINT "Your reaction time is ";
PRINT endTime - startTime; " milliseconds"
END
```

MID$

Purpose

Return characters from the middle of a string.

Syntax

MID$(*string$*, start, length)

Description

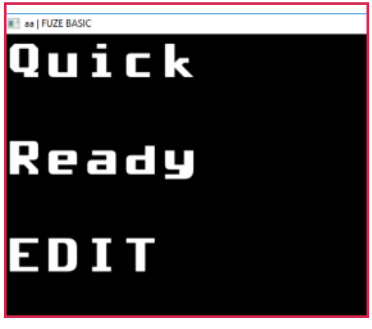
Returns the middle *length* characters of *string$* starting from position *start*. The first character of the string is position number 0.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
REM Prints Quick
CLS
FONTSIZE ( 6 )
string$ = "The Quick Brown Fox"
PRINT MID$( string$, 4, 5 )
END
```



MIDDLECLICK

Purpose

Returns TRUE if the middle mouse button is clicked

Syntax

value = MIDDLECLICK

Description

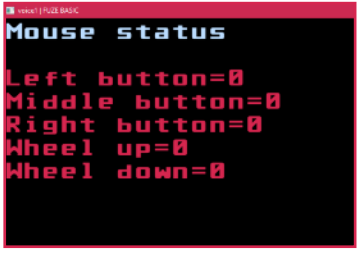
Returns a value of 0 or 1 (TRUE or FALSE) depending on whether it is press or not.

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
UPDATEMODE = 0
FONTSIZE (4)
LOOP
 CLS2
 INK = LIGHTBLUE
 PRINTAT (0, 0);  "Mouse status"
 INK = RASPBERRY
 PRINTAT (0, 2);  "Left button=";  LEFTCLICK
 PRINTAT (0, 3);  "Middle button=";  MIDDLECLICK
 PRINTAT (0, 4);  "Right button=";  RIGHTCLICK
 PRINTAT (0, 5);  "Wheel up=";  WHEELUP
 PRINTAT (0, 6);  "Wheel down=";  WHEELDOWN
 UPDATE
REPEAT
```



MIN

Purpose

Returns the smaller of two numbers.

Syntax

minvalue = MIN(*number1*, *number2*)

Description

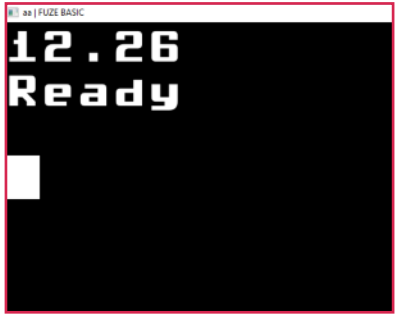
Returns the smaller (lowest value) of *number1* or *number2*.

Associated

MAX

Example

```
REM Prints the value of number1
number1 = 12.26
number2 = 27.45
PRINT MIN( number1, number2 )
END
```



MOUSEOFF / MOUSEON

Purpose

Make the mouse cursor visible and invisible.

Syntax

MOUSEOFF
MOUSEON

Description

Make the mouse cursor invisible or visible within the FUZE Code Studio window. Off is the default value.

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
CLS
MOUSEON
LOOP
   GETMOUSE( x, y, b )
   CIRCLE( x, y, 10, RND(255) )
   UPDATE
   REM LOOP colour if left button pressed
   IF b = 1 THEN
      COLOUR = COLOUR MOD 30 + 1
   ENDIF
   REM Exit if right button pressed
REPEAT UNTIL b = 4
MOUSEOFF
END
```

MOUSEX / MOUSEY

Purpose

To find the mouse X position

Syntax

value1 = MOUSEX
value2 = MOUSEY

Description

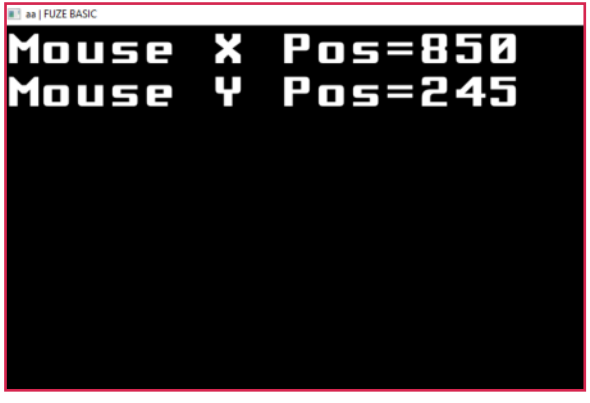
Returns the X or Y position of the current mouse location

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
CLS
UPDATEMODE = 0
FONTSIZE ( 5 )
LOOP
CLS2
PRINTAT( 0, 0 ); "Mouse X Position="; MOUSEX;
PRINTAT( 0, 1 ); "Mouse Y Position="; MOUSEY;
UPDATE
REPEAT
```



MOUSEBUTTON

Purpose

Read the button(s) state from an attached mouse

Syntax

value = MOUSEBUTTON

Description

Read the current state of the mouse buttons.

The returned values are;

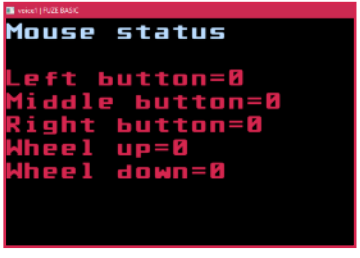
0 - no buttons pressed
1 - left button pressed
2 - middle button pressed
3 - left & middle buttons pressed
4 - right button pressed
5 - left & right buttons pressed
6 - right & middle button pressed
7 - left, middle & right buttons pressed

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
UPDATEMODE = 0
FONTSIZE (4)
LOOP
 CLS2
 INK = LIGHTBLUE
 PRINTAT (0, 0);  "Mouse status"
 INK = RASPBERRY
 PRINTAT (0, 2);  "Button value=";  MOUSEBUTTON
 UPDATE
REPEAT
```



MOVE

Purpose

Move the graphics turtle forward

Syntax

MOVE(*distance*)

Description

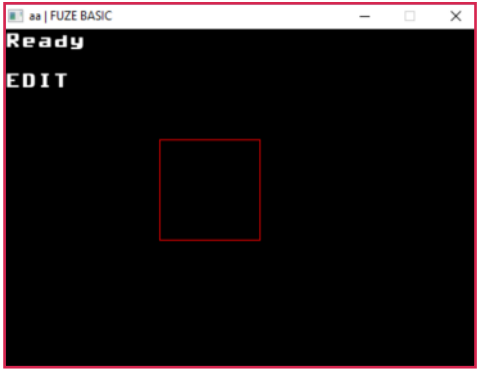
This causes the virtual graphics turtle to move forwards *distance* in screen pixels. A line will be drawn if the pen is down.

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE

Example

```
CLS
COLOUR = RED
MOVE( 50 )
DEG
LEFT( 90 )
MOVE( 50 )
PENDOWN
FOR I = 1 TO 4 LOOP
   LEFT( 90 )
   MOVE( 100 )
REPEAT
UPDATE
END
```



MOVETO

Purpose

Move the graphics turtle to a point on the screen.

Syntax

MOVETO(*xpos*, *ypos*)

Description

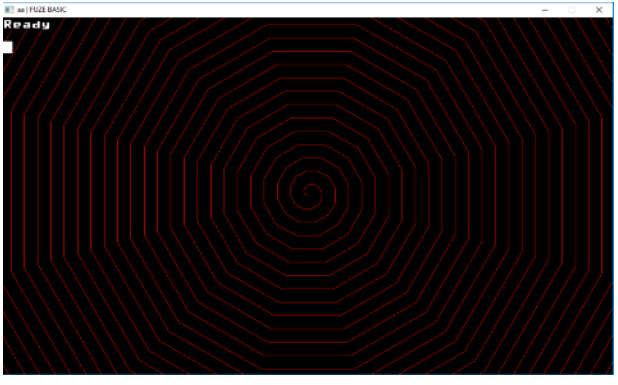
This moves the virtual graphics turtle to the absolute location (*xpos*, *ypos*). A line will be drawn if the pen is down.

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE

Example

```
REM Draw a spiral in the centre of the screen
CLS
COLOUR = RED
PENUP
MOVETO( GWIDTH / 2, GHEIGHT / 2 )
PENDOWN
FOR I = 2 TO GWIDTH LOOP
   MOVE( I )
   RIGHT( 30 )
REPEAT
UPDATE
END
```



NEWSPRITE

Purpose

Create a new sprite.

Syntax

index = NEWSPRITE(*count*)

Description

This returns an index (or handle) to the internal sprite data. You need to use the index returned in all future sprite handling functions/procedures. The *count* argument specifies the number of different versions of the sprite.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
COLOUR = YELLOW
CIRCLE( 100, 100, 50, TRUE )
SAVEREGION( "pac1.bmp", 50, 50, 100, 100 )
COLOUR = BLACK
TRIANGLE( 100, 100, 150, 125, 150, 75, TRUE )
SAVEREGION( "pac2.bmp", 50, 50, 100, 100 )
TRIANGLE( 100, 100, 150, 150, 150, 50, TRUE )
SAVEREGION( "pac3.bmp", 50, 50, 100, 100 )
CLS
S = 0
pacman = NEWSPRITE( 4 )
LOADSPRITE( "pac1.bmp", pacman, 0 )
LOADSPRITE( "pac2.bmp", pacman, 1 )
LOADSPRITE( "pac3.bmp", pacman, 2 )
LOADSPRITE( "pac2.bmp", pacman, 3 )
FOR X = 1 TO GWIDTH STEP 25 LOOP
    PLOTSPRITE( pacman, X, GHEIGHT / 2, S )
    S = S + 1
    IF S = 4 THEN S = 0
    WAIT( 0.05 )
    UPDATE
REPEAT
HIDESPRITE( pacman )
END
```

NUMFORMAT

Purpose

Control how numbers are formatted.

Syntax

NUMFORMAT(*width*, *decimals*)

Description

You can affect the way numbers are printed using the NUMFORMAT procedure. This takes 2 arguments, the *width* specifying the total number of characters to print and the *decimals* the number of characters after the decimal point. Numbers printed this way are right-justified with leading spaces inserted if required. NUMFORMAT (0,0) restores the output to the general purpose format used by default.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
NUMFORMAT( 6, 4 )
REM Prints 3.1416
PRINT PI
NUMFORMAT( 0, 0 )
REM Prints 3.141592654
PRINT PI
END
```

OPEN

Purpose

Open a file for read or write.

Syntax

handle = OPEN(*filename$*)

Description

The OPEN function opens a file and makes it available for reading or writing and returns the numeric *handle* associated with the file. The file is created if it doesn't exist, or if it does exist the file pointer is positioned at the start of the file.

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN( "testfile.txt" )
PRINT# handle, "Colin"
PRINT# handle, 47
CLOSE( handle )
handle = OPEN( "testfile.txt" )
INPUT# handle, Name$
INPUT# handle, Age
CLOSE( handle )
PRINT "Name: " + Name$
PRINT "Age: "; Age
END
```

ORIGIN

Purpose

Move the graphics origin.

Syntax

ORIGIN(*xpos*, *ypos*)

Description

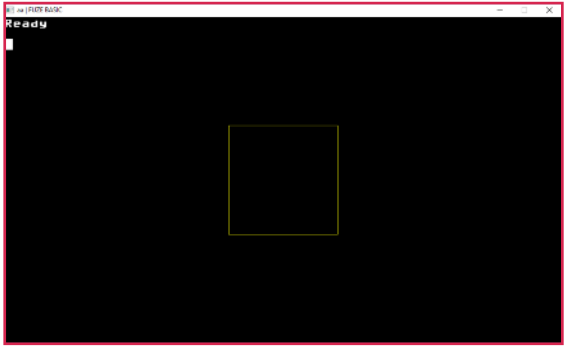
This changes the graphics origin for the Cartesian plotting procedures. The *xpos*, *ypos* coordinates are always absolute coordinates with (0,0) being bottom left (the default).

Associated

FULLSCREEN, GHEIGHT, GWIDTH, ORIGIN, SETMODE

Example

```
CLS
COLOUR = YELLOW
REM Move the origin to the screen centre
ORIGIN( GWIDTH / 2, GHEIGHT / 2 )
PLOT( -100, -100 )
LINETO( -100, 100 )
LINETO( 100, 100 )
LINETO( 100, -100 )
LINETO( -100, -100 )
UPDATE
ORIGIN( 0, 0 )
END
```



PAPER

Purpose

Set/Read the current text background colour.

Syntax

backgroundcolour = PAPER

PAPER = backgroundcolour

Description

Set/Read the current text background (paper) colour. Clear screen (CLS) will set the entire background to this colour.

Example

```
PRINT "Text background colour "; PAPER
PAPER = RED
PRINT "Text background colour "; PAPER
PAPER = 7
PRINT "Text background colour "; PAPER
END
```

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

PAPEROFF / PAPERON

Purpose

Switches the text background colour off.

Syntax

PAPEROFF
PAPERON

Description

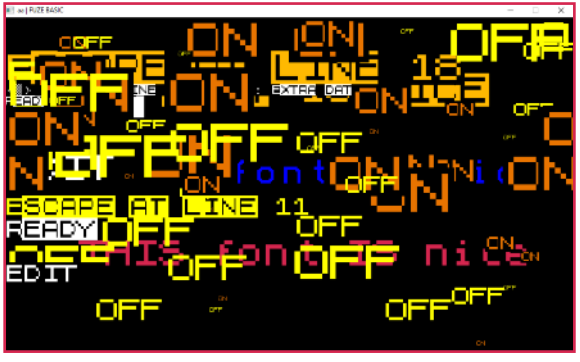
This function switches the background text colour off or on. It can be turned on or off so that the text background does not obscure whatever is behind it.

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

Example

```
LOOP
  INK = ORANGE
  PAPERON
  FONTSIZE ( RND ( 10 ) + 1 )
  PRINTAT ( RND( TWIDTH), RND( THEIGHT - 1 )); "ON";
  WAIT( 0.3 )
  INK = YELLOW
  PAPEROFF
  FONTSIZE ( RND ( 10 ) + 1 )
  PRINTAT ( RND( TWIDTH ), RND( THEIGHT - 1 )); "OFF";
  WAIT( 0.3 )
REPEAT
```



PAUSECHAN

Purpose

Pause the playing of a sound sample.

Syntax

PAUSECHAN(*handle*)

Description

This function pauses the playing of the sound sample associated with the *handle* returned by *LOADSAMPLE* that has been started using *PLAYSAMPLE*. The sample can be resumed where it left off using *RESUMECHAN*

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
channel = 0
volume = 70
SETCHANVOL( channel, volume )
intro = LOADSAMPLE( "parClip6.wav" )
PLAYSAMPLE( intro, channel, 0 )
WAIT( .3 )
PAUSECHAN( intro )
WAIT( .3 )
RESUMECHAN( intro )
END
```

PAUSEMUSIC

Purpose

Pause a playing music file

Syntax

PAUSEMUSIC

Description

Pauses a playing music track which can them be restarted using RESUMEMUSIC.

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
handle = LOADMUSIC( "neon.mp3" )
SETMUSICVOL( 70 )
PLAYMUSIC( handle, 0 )
WAIT(1)
PAUSEMUSIC
WAIT(1)
PLAYMUSIC( handle, 0 )
END
```

PENDOWN / PENUP

Purpose

Start / stop drawing using the graphics turtle.

Syntax

PENDOWN
PENUP

Description

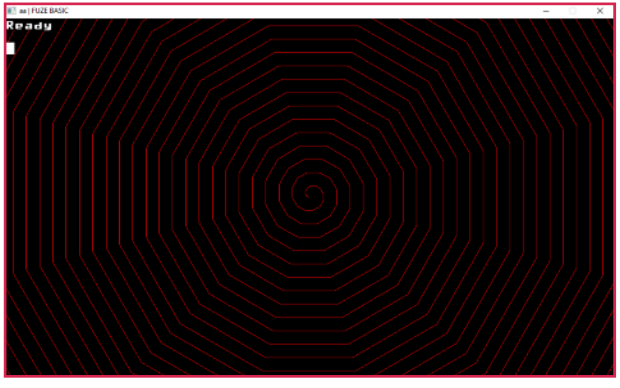
This lowers or raises the virtual graphics turtle "pen". Nothing will be drawn until you execute this procedure.

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE

Example

```
REM Draw a spiral in the centre of the screen
CLS
COLOUR = RED
PENUP
MOVETO( GWIDTH / 2, GHEIGHT / 2 )
PENDOWN
FOR I = 2 TO GWIDTH LOOP
   MOVE( I )
   RIGHT( 30 )
REPEAT
UPDATE
END
```



PI / PI2

Purpose

Returns the value of the constant Pi and Pi divided by 2

Syntax

valueofpi = PI
valueofpi = PI2 // divided by 2

Description

Returns an approximation of the value of the constant pi which is the ratio of a circle's circumference to its diameter (approximately 3.14159265) which is widely used in mathematics, specifically trigonometry and geometry.

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
PRINT FN AreaOfCircle( 12 )
END

DEF FN AreaOfCircle( withRadius )
   LOCAL result
   result = PI * withRadius * withRadius
= result
```

PINMODE

Purpose

Configure the mode of a pin on the Pi's GPIO.

Syntax

PINMODE(*pinno*, *pinmode*)

Description

Configures the mode of a pin on the Pi's GPIO. It takes an argument which specifies the mode of the pin - input, output or PWM output. The modes are:

0 pinINPUT
1 pinOUTPUT
2 pinPWM (FUZE / Rpi)
2 pinINPUT with pull-up resistors enabled (Arduino)

Note:

If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

ANALOGREAD, ANALOGWRITE, DIGITALREAD, DIGITALWRITE, PINMODE, PWMWRITE, SOFTPWMWRITE

Example

```
REM Set pin 2 to input
PINMODE( 2, 1 )
REM Wait for button to be pushed
UNTIL DIGITALREAD( 2 ) = TRUE LOOP
REPEAT
PRINT "Button Pushed"
END
```

PLAYMUSIC

Purpose

Start playing a music track.

Syntax

PLAYMUSIC(*handle*, *repeats*)

Description

This function plays a music track previously loaded using the *LOADMUSIC* function which returns the *handle*. The *repeats* are the number of times to play the track.

Example

```
handle = LOADMUSIC( "neon.mp3" )
SETMUSICVOL( 70 )
PLAYMUSIC( handle, 1 )
END
```

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

PLAYSAMPLE

Purpose

Start playing a sound sample.

Syntax

PLAYSAMPLE(*handle*, *channel*, *loops*)

Description

This function plays a sound previously loaded using the *LOADSAMPLE* function which returns the *handle*. The *channel* is 0, 1, 2 or 3 which lets you play up to 4 concurrent samples. The *loops* parameter is different to the *repeats* one in the *PLAYMUSIC* function. Here it means the number of times to loop the sample – zero means no loops which means play it once.

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
channel = 0
volume = 70
SETCHANVOL( channel, volume )
intro = LOADSAMPLE( "parClip6.wav" )
PLAYSAMPLE( intro, channel, 0 )
END
```

PLOT

Purpose

Draw a single point on the screen.

Syntax

PLOT(*xpos*, *ypos*)

Description

This plots a single pixel at screen location (*xpos*,*ypos*) in the selected graphics mode in the selected colour. Note that (0,0) is bottom left by default.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
LOOP
   IF INKEY <> -1 THEN BREAK
   xpos = RND( GWIDTH )
   ypos = RND( GHEIGHT )
   COLOUR = RND( 30 )
   PLOT( xpos, ypos )
   UPDATE
REPEAT
END
```

PLOTIMAGE

Purpose

Display a loaded image on the screen.

Syntax

PLOTIMAGE(*handle*, *xpos*, *ypos*)

Description

Plot an image previously loaded using LOADIMAGE (using the *handle* returned by LOADIMAGE) on the screen at coordinates (*xpos*,*ypos*).

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
COLOUR = RED
RECT( 0, 0, 50, 50, TRUE )
COLOUR = WHITE
LINE( 0, 50, 50, 50 )
LINE( 0, 25, 50, 25 )
LINE( 0, 25, 0, 50 )
LINE( 50, 25, 50, 50 )
LINE( 25, 0, 25, 25 )
LINE( 0, 0, 50, 0 )
SAVEREGION( "bricks.bmp", 0, 0, 50, 50 )
handle = LOADIMAGE( "bricks.bmp" )
FOR X = 0 TO GWIDTH STEP 50 LOOP
  FOR Y = 0 TO GHEIGHT STEP 50 LOOP
    PLOTIMAGE( handle, X, Y )
  REPEAT
REPEAT
UPDATE
END
```

PLOTSPRITE

Purpose

Draw a sprite on the screen.

Syntax

PLOTSPRITE(*index*, *xpos*, *ypos*, *subindex*)

Description

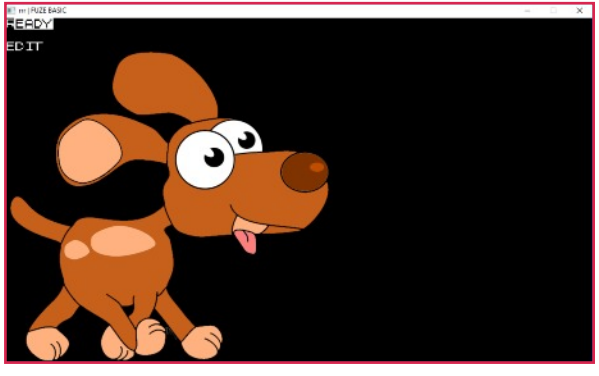
This plots the sprite *index* and version *subindex* at the coordinates (*xpos*, *ypos*). The coordinates specify the bottom-left corner of the bounding rectangle of the sprite.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
REM Create a new sprite with 1 version
SpriteIndex = NEWSPRITE( 1 )
REM Load a sprite from a file
fuzelogo$ = "dog1.png"
LOADSPRITE( fuzelogo$, SpriteIndex, 0 )
REM Draw the sprite on the screen
PLOTSPRITE( SpriteIndex, 0, 0, 0 )
UPDATE
END
```



POLYEND / POLYSTART

Purpose

Indicates the start and end of a filled polygon.

Syntax

POLYEND
POLYSTART

Description

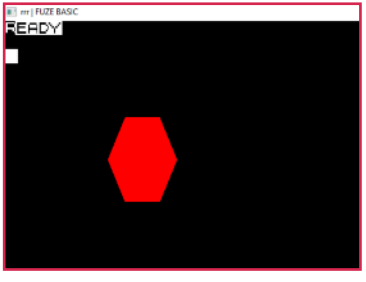
This marks the beginning and end of a polygon function. The PLOYPLOT statements within plot the individual points of the polygon to be drawn.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
PROC Hexagon( 200, 400, 50, Red )
UPDATE
END
DEF PROC Hexagon( x, y, l, c )
   COLOUR = c
   POLYSTART
   POLYPLOT( x + l, y )
   POLYPLOT( x + l / 2, y - l * SQRT( 3 / 2 ))
   POLYPLOT( x - l / 2, y - l * SQRT( 3 / 2 ))
   POLYPLOT( x - l, y )
   POLYPLOT( x - l / 2, y + l * SQRT( 3 / 2 ))
   POLYPLOT( x + l / 2, y + l * SQRT( 3 / 2 ))
   POLYEND
ENDPROC
```



POLYPLOT

Purpose

Add a point to a filled polygon.

Syntax

POLYPLOT(*xpos*, *ypos*)

Description

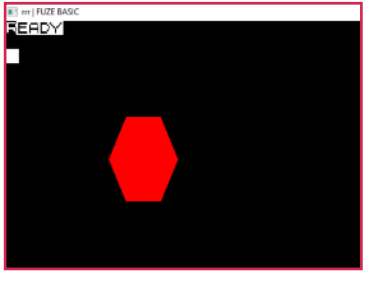
This remembers the given *xpos*,*ypos* coordinates as part of a filled polygon. Nothing is actually drawn on the screen until the POLYEND instruction is executed. Polygons can have a maximum of 64 points.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
PROC Hexagon( 200, 400, 50, Red )
UPDATE
END
DEF PROC Hexagon( x, y, l, c )
  COLOUR = c
  POLYSTART
  POLYPLOT( x + l, y )
  POLYPLOT( x + l / 2, y - l * SQRT( 3 / 2 ))
  POLYPLOT( x - l / 2, y - l * SQRT( 3 / 2 ))
  POLYPLOT( x - l, y )
  POLYPLOT( x - l / 2, y + l * SQRT( 3 / 2 ))
  POLYPLOT( x + l / 2, y + l * SQRT( 3 / 2 ))
  POLYEND
ENDPROC
```



PLOTTEXT

Purpose

Display text using graphic coordinates.

Syntax

PLOTTEXT(“text”, xpos, ypos)

Description

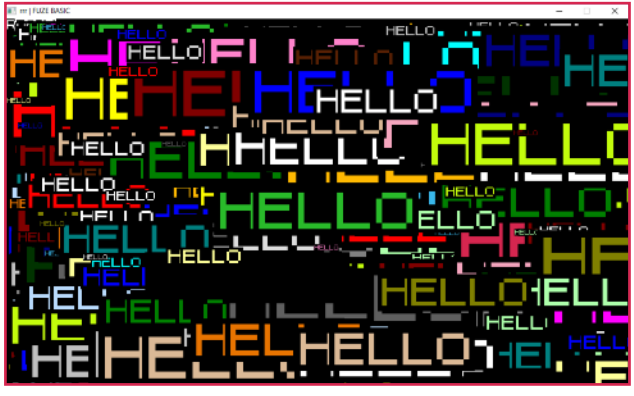
The PRINT command uses the cursor coordinates to display text whereas PLOTTEXT can position text at a specified pixel location

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
LOOP
INK = RND ( 30 )
FONTSIZE ( RND ( 10 ) + 1 )
PLOTTEXT( "HELLO", RND( GWIDTH ), RND( GHEIGHT - 1 ))
UPDATE
REPEAT
```



PRINT

Purpose

Output text to the screen

Syntax

PRINT "*text*" { ; "*text*" }

Description

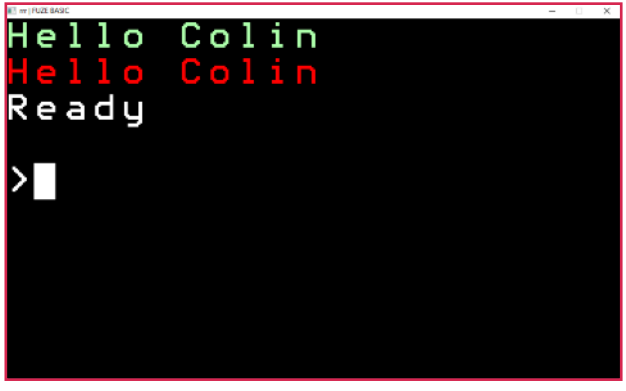
Outputting text to the screen is done via the PRINT command. The PRINT command is quite versatile and will print any combination of numbers and strings separated by the semi-colon (;). A trailing semi-colon will suppress the printing of a new line.

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

Example

```
CLS
FONTSIZE ( 6 )
INK = LIGHTGREEN
PRINT "Hello Colin"
INK = RED
name$ = "Colin"
PRINT "Hello ";
PRINT name$
END
```



PRINT#

Purpose

Print data to a file.

Syntax

PRINT# *handle*, *data*

Description

The PRINT# instruction acts just like the regular PRINT instruction except that it sends data to the file identified by the supplied file-handle rather than to the screen. Numbers are formatted according to the settings of NUMFORMAT. It is strongly recommended to only print one item per line if you are going to read those items back into a FUZE BASIC program again.

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN( "testfile.txt" )
PRINT# handle, "Hello World"
CLOSE ( handle )
END
```

PRINTAT

Purpose

Set the text cursor position and print

Syntax

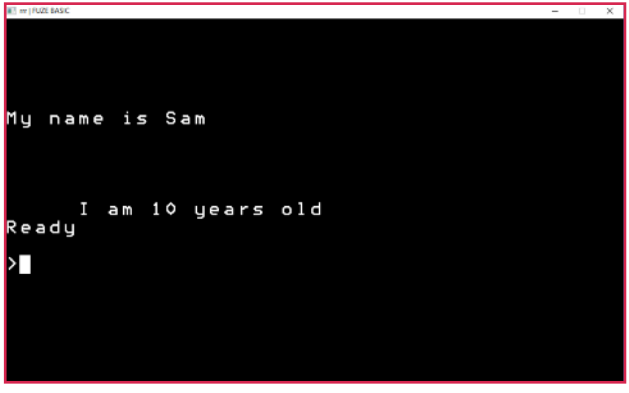
PRINTAT(*x*, *y*); "*text*"

Description

Use to position the text cursor at the specified location and print. Useful for laying out text and or variables at any preferred screen location.

Example

```
CLS
FONTSIZE ( 3 )
name$ = "Sam"
age = 10
PRINTAT( 0, 5 ); "My name is "; name$
PRINTAT( 5, 10 ); "I am "; age ; " years old"
END
```



Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

PROC

Purpose

Call a user defined procedure.

Syntax

PROC *name*({*argument*} { , *argument* })

Description

Calls the specified user defined procedure called *name* with the specified *arguments*. Once the procedure has been executed control returns to the command following.

Associated

DEF PROC, LOCAL, PROC, ENDPROC

Example

```
CLS
FONTSIZE ( 4 )
LOOP
   x = RND( TWIDTH )
   y = RND( THEIGHT )
   c = RND( 30 )
   text$ = "Blossom"
   PROC text( text$, x, y, c )
REPEAT
END
DEF PROC text( text$, x, y, c )
 INK = c
 PRINTAT( x, y ); text$
ENDPROC
```

PWMWRITE

Purpose

Output a PWM waveform on the selected pin.

Syntax

PWMWRITE(*pinno*, *pinvalue*)

Description

This procedure outputs a PWM waveform on the selected pin. The pin must be configured for PWM mode beforehand. The value set should be between 0 and 100.

NOTE: This is applicable to FUZE & Rpi only. For Arduino see Analogwrite.

Example

```
REM Set pin 1 to PWM output mode
PINMODE( 1, 2 )
PWMWRITE ( 1, 50 )
END
```

Associated

ANALOGREAD, ANALOGWRITE, DIGITALREAD, DIGITALWRITE, PINMODE, PWMWRITE, SOFTPWMWRITE

RAD

Purpose

Set angle units to radians.

Syntax

RAD

Description

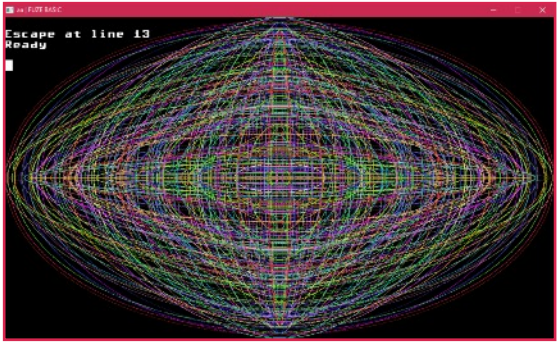
Switches the internal angle system to radians. There are 2 * PI radians in a full circle.

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
REM Draw an ellipse in the screen centre
CLS
RAD
LOOP
width=RND(GWIDTH/2)
height=RND(GHEIGHT/2)
RGBCOLOUR (RND(255), RND(255), RND(255))
FOR angle = 0 TO 6 * PI STEP 0.01 LOOP
xpos = width * COS( angle ) + GWIDTH / 2
ypos = height * SIN( angle ) + GHEIGHT / 2
PLOT( xpos, ypos )
REPEAT
UPDATE
REPEAT
END
```



READ

Purpose

Read data into program variables.

Syntax

READ *variable* { , *variable* }

Description

To get data into your program variables, we use the READ instruction. We can read one, or many items of data at a time.

Associated

DATA, READ, RESTORE

Example

```
REM Load the name of the days of the
REM week into a string array
DATA "Monday", "Tuesday", "Wednesday"
DATA "Thursday", "Friday", "Saturday"
DATA "Sunday"
DIM daysofweek$( 7 )
FOR day = 1 TO 7 LOOP
   READ daysofweek$( day )
REPEAT
PRINT "The third day of the week is ";
PRINT daysofweek$( 3 )
END
```

RECT

Purpose

Draw a rectangle on the screen.

Syntax

RECT(*xpos*, *ypos*, *width*, *height*, *fill*)

Description

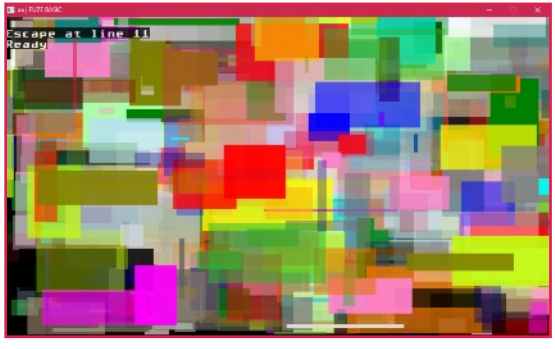
Draws a rectangle at position (*xpos*,*ypos*) with *width* and *height*. The final parameter, *fill* is either TRUE, FALSE.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
LOOP
   COLOUR = RND( 30 )
   x = RND( GWIDTH )
   y = RND( GHEIGHT )
   w = RND( GWIDTH / 4 )
   h = RND( GHEIGHT / 4 )
   SETALPHA (RND( 255 ))
   RECT( x, y, w, h, 1 )
   UPDATE
   IF INKEY <> -1 THEN BREAK
REPEAT
END
```



REPEAT UNTIL

Purpose

Loop until the specified condition is met.

Syntax

LOOP
  { *statements* }
REPEAT UNTIL *condition*

Description

Execute the *statements* one or more times until the *condition* is TRUE (not 0).

Example

```
number = INT( RND( 10 )) + 1
guess = 0
REM Guessing Game
PRINT "Guess a Number Between 1 and 10"
LOOP
  INPUT "Enter Your Guess: ", guess
  IF ( number <> guess ) THEN
    PRINT "Incorrect Guess Again"
  ENDIF
REPEAT UNTIL number = guess
PRINT "You are Correct!"
END
```

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT UNTIL, UNTIL REPEAT, WHILE REPEAT

RESTORE

Purpose

Reset the DATA pointer.

Syntax

RESTORE

Description

Resets the READ command to the very first DATA statement in the program.

Associated

DATA, READ, RESTORE

Example

```
DATA "Monday", "Tuesday", "Wednesday"
DATA "Thursday", "Friday", "Saturday", "Sunday"
FOR day = 1 TO 3 LOOP
  READ dayofweek$
REPEAT
PRINT dayofweek$
RESTORE
FOR day = 1 TO 4 LOOP
  READ dayofweek$
REPEAT
PRINT dayofweek$
END
```

RESUMECHAN

Purpose

Resume the playing of a sound sample.

Syntax

RESUMECHAN(*handle*)

Description

This function resumes the playing of the sound sample associated with the *handle* returned by *LOADSAMPLE* that has been started using *PLAYSAMPLE* and paused using *PAUSECHAN*.

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
channel = 0
volume = 70
SETCHANVOL( channel, volume )
intro = LOADSAMPLE( "parClip6.wav" )
PLAYSAMPLE( intro, channel, 0 )
WAIT( 3 )
PAUSECHAN( intro )
WAIT( 2 )
RESUMECHAN( intro )
END
```

RESUMEMUSIC

Purpose

Resumes music playing after it has been paused.

Syntax

RESUMEMUSIC

Description

Resumes the playing of a music track previously paused using PAUSEMUSIC.

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
handle = LOADMUSIC( "ForNext.mp3" )
SETMUSICVOL( 70 )
PLAYMUSIC( handle, 1 )
WAIT( 2 )
PAUSEMUSIC
WAIT( 1 )
RESUMEMUSIC
END
```

REWIND

Purpose

Move the file pointer to the start of a file.

Syntax

REWIND(*handle*)

Description

Move the file pointer to the start of the file specified by *handle*.

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN ( "rewindtest.txt" )
PRINT# handle, "First Record"
PRINT# handle, "Second Record"
CLOSE( handle )
handle = OPEN( "rewindtest.txt" )
INPUT# handle, record$
PRINT record$
REWIND( handle )
REM reads the first record again
INPUT# handle, record$
PRINT record$
CLOSE ( handle )
END
```

RGBCOLOUR

Purpose

Set the current graphical plot colour to an RGB (Red,Green, Blue) value.

Syntax

RGBCOLOUR(*red*, *green*, *blue*)

Description

This sets the current graphical plot colour to an RGB (Red,Green, Blue) value. The values should be from 0 to 255.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
PRINT "Draw Spectrum"
FOR value = 0 TO 255 LOOP
RGBCOLOUR( 255, value,  0 )
LINE( value , 0, value , GHEIGHT )
RGBCOLOUR( value , 255, 0 )
LINE( 511 - value , 0, 511 - value , GHEIGHT )
RGBCOLOUR( 0, 255 - value , value  )
LINE( 512 + value , 0, 512 + value , GHEIGHT )
RGBCOLOUR( 0, value , 255 )
LINE( 768 + value , 0, 768 + value , GHEIGHT )
REPEAT
UPDATE
END
```

RIGHT

Purpose

Turns the turtle to the right (clockwise) by the given angle.

Syntax

RIGHT(*angle*)

Description

Turns the virtual graphics turtle to the right (clockwise) by the given *angle* in the current angle units.

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE

Example

```
CLS
PRINT "Draw Pink Hexagon"
PENUP
COLOUR = PINK
PENDOWN
FOR I = 1 TO 6 LOOP
   RIGHT( 60 )
   MOVE( 100 )
REPEAT
END
```

RIGHT$

Purpose

Return the specified rightmost number of a characters from a string.

Syntax

*substring$* = RIGHT$(*string$*, *number*)

Description

Returns a substring of *string$* with *number* characters from the right (end) of the string. If number is greater than or equal to the length of *string$* then the whole string is returned.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
CLS
FONTSIZE( 3 )
string$ = "The quick brown fox"
FOR i = 1 TO 20 LOOP
   PRINT RIGHT$( string$, i )
REPEAT
END
```

RIGHTCLICK

Purpose

Returns TRUE if the right mouse button is clicked

Syntax

value = RIGHTCLICK

Description

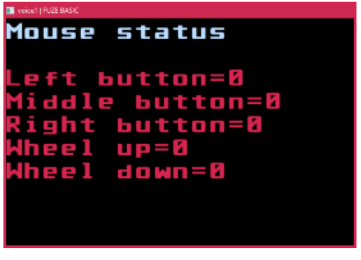
Returns a value of 0 or 1 (TRUE or FALSE) depending on whether it is press or not.

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
UPDATEMODE = 0
FONTSIZE (4)
LOOP
 CLS2
 INK = LIGHTBLUE
 PRINTAT (0, 0);  "Mouse status"
 INK = RASPBERRY
 PRINTAT (0, 2);  "Left button=";  LEFTCLICK
 PRINTAT (0, 3);  "Middle button=";  MIDDLECLICK
 PRINTAT (0, 4);  "Right button=";  RIGHTCLICK
 PRINTAT (0, 5);  "Wheel up=";  WHEELUP
 PRINTAT (0, 6);  "Wheel down=";  WHEELDOWN
 UPDATE
REPEAT
```



ROTATEIMAGE

Purpose

To rotate an image by a specified number of degrees.

Syntax

ROTATEIMAGE(handle, degrees)

Description

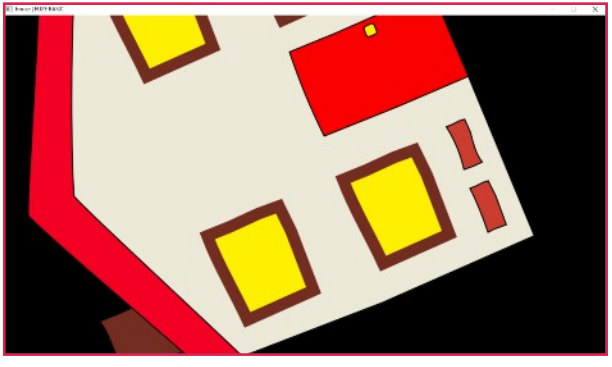
Use to rotate the specified image by the number of degrees specified.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
CLS
image = LOADIMAGE ( "house3.png" )
angle=0
LOOP
CLS
ROTATEIMAGE ( image, angle )
PLOTIMAGE ( image, 0, 0 )
angle=angle+22.5
UPDATE
WAIT ( .3 )
REPEAT
END
```



RND

Purpose

Generate a random number in a given range.

Syntax

random = RND(*range*)

Description

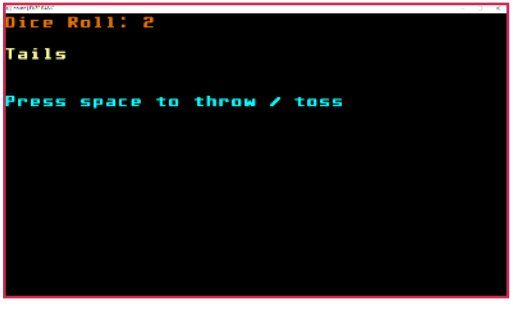
This function returns a random number based on the value of *range*. If *range* is zero, then the last random number generated is returned, if *range* is 1, then a random number from 0 to 1 is returned, otherwise a random number from 0 up to, but not including *range* is returned.

Associated

SEED

Example

```
CLS
FONTSIZE (4)
LOOP
  diceroll = RND (6) + 1
  INK = ORANGE
  PRINTAT (0, 0);  "Dice Roll: ";  diceroll
  cointoss = RND (2)
  INK = LIGHTYELLOW
  IF cointoss = 1 THEN PRINTAT (0, 2);  "Heads"
  IF cointoss = 0 THEN PRINTAT (0, 2);  "Tails"
  WHILE INKEY <> 32 LOOP
    INK = RND (30)
    PRINTAT (0, 5);  "Press space to throw / toss"
  REPEAT
REPEAT
END
```



SAVEREGION

Purpose

Save a snapshot of an area of the screen to an image file.

Syntax

SAVEREGION(*file$*, *xpos*, *ypos*, *width*, *height*)

Description

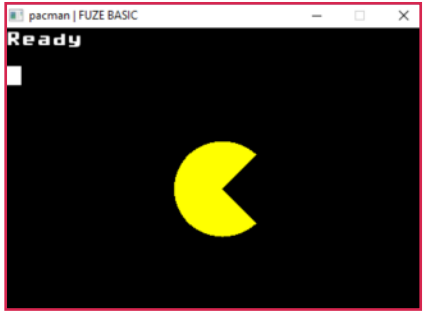
Takes a snapshot of an area of the current screen specified by the rectangle with bottom left at coordinates (*xpos*,*ypos*) of specified *width* and *height*, and saves it to the file named *file$* in a bitmap (.bmp) format.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
CLS
COLOUR = YELLOW
CIRCLE ( 100, 100, 50, TRUE )
COLOUR = BLACK
TRIANGLE( 100, 100, 150, 150, 150, 50, TRUE )
TRIANGLE( 100, 100, 150, 125, 150, 75, TRUE )
SAVEREGION( "pac1.bmp", 50, 50, 100, 100 )
CLS
pacman = NEWSPRITE( 1 )
LOADSPRITE( "pac1.bmp", pacman , 0 )
FOR x = 1 TO GWIDTH STEP 25 LOOP
PLOTSPRITE( pacman, x, GHEIGHT / 2, 0 )
UPDATE
WAIT( 0.1 )
REPEAT
```



SAVESCREEN

Purpose

Save a snapshot of the screen to an image file.

Syntax

SAVESCREEN(*filename$*)

Description

This takes a snapshot of the current screen and saves it to the filename given in a bitmap (.bmp) format file.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
CLS
PRINT "Draw Spectrum"
FOR value = 0 TO 255 LOOP
RGBCOLOUR( 255, value,  0 )
LINE( value , 0, value , GHEIGHT )
RGBCOLOUR( value , 255, 0 )
LINE( 511 - value , 0, 511 - value , GHEIGHT )
RGBCOLOUR( 0, 255 - value , value  )
LINE( 512 + value , 0, 512 + value , GHEIGHT )
RGBCOLOUR( 0, value , 255 )
LINE( 768 + value , 0, 768 + value , GHEIGHT )
REPEAT
UPDATE
SAVESCREEN( "screenshot.bmp" )
END
```

SCALEIMAGE

Purpose

Resize a loaded image.

Syntax

SCALEIMAGE(*handle, percent*)

Description

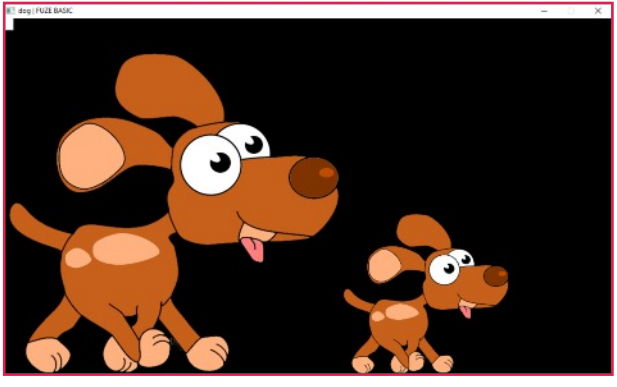
Scales a preloaded image by a specified percentage.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
CLS
image = LOADIMAGE ("dog1.png")
PLOTIMAGE (image, 0, 0)
newx = GETIMAGEW (image)
UPDATE
WAIT (1)
SCALEIMAGE (image, 50)
PLOTIMAGE (image, newx, 0)
UPDATE
WAIT (1)
END
```



SCANKEYBOARD

Purpose

Scan for a key pressed down.

Syntax

SCANKEYBOARD(*keycode*)

Description

Allows you to detect that any of the keys have been pressed (including special keys) and also to detect mutiple keys pressed at the same time. The *keycode* parameter indicates the key press to be scanned for e.g scanSpace is the space bar. See the end of this guide for a full list of SCANKEYBOARD keycodes.

Associated

CLEARKEYBOARD, GET, GET$, INKEY, INPUT, SCANKEYBOARD

Example

```
PRINT "Press Ctrl-Alt-P"
LOOP
  lctrl = SCANKEYBOARD( scanLCtrl )
  lalt  = SCANKEYBOARD( scanLAlt )
  p = SCANKEYBOARD( scanP )
  reboot = lctrl AND lalt AND p
REPEAT UNTIL reboot
PRINT "Rebooting..."
CLEARKEYBOARD
END
```

SCLOSE

Purpose

Close an open serial port.

Syntax

SCLOSE(*handle*)

Description

This closes a serial port and frees up any resources used by it. It's not strictly necessary to do this when you end your program, but it is considered good practice.

Associated

SCLOSE, SGET, SGET$, SOPEN, SPUT, SPUT$, SREADY

Example

```
REM Read a character from a serial port
arduino = SOPEN( "/dev/ttyUSB0", 115200 )
char$ = SGET$( arduino )
PRINT char$
SCLOSE( arduino )
END
```

SCROLLDOWN / SCROLLLEFT SCROLLRIGHT / SCROLLUP

Purpose

Scroll a region of the screen up, down, left or right.

Syntax

SCROLLDOWN(*xpos*, *ypos*, *width*, *height*, *pixels*)

Description

Scroll the screen region specified by *xpos* , *ypos, width & height* by the given number of *pixels*.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example


```
CLS
w = 100 // Width
s = 2 // Step Size
x = ( GWIDTH - w ) / 2
y = ( GHEIGHT - w ) / 2
COLOUR = WHITE
RECT( x, y, w, w, TRUE )
RECT( x + w, y + w, w, w, TRUE )
RECT( x - 2, y - 2, w * 2 + 4, w * 2 + 4, FALSE )
UPDATE
COLOUR = BLACK
FOR i = 1 TO w STEP s LOOP
   SCROLLUP( x, y, w, w * 2, s )
   SCROLLDOWN( x + w, y, w, w * 2, s )
   UPDATE
   WAIT( 0.01 )
REPEAT
FOR i = 1 TO w STEP s LOOP
   SCROLLRIGHT( x, y + w, w * 2, w, s )
   SCROLLLEFT( x, y, w * 2, w, s )
   WAIT( 0.01 )
UPDATE
REPEAT
END
```

SEED

Purpose

Seed the random number generator.

Syntax

SEED = *value*

Description

This can be assigned to initialise the random number generator, or you can read it to find the current seed.

Associated

RND

Example

```
SEED = 10
PRINT RND( 100 )
SEED = 10
REM Will print the same number
PRINT RND( 100 )
REM Will print a different number
PRINT RND( 100 )
END
```

SEEK

Purpose

Move the file pointer to any place in the file.

Syntax

SEEK(*handle*, *offset*)

Description

The SEEK instruction moves the file pointer to any place in the file. It can even move the file pointer beyond the end of the file in which case the file is extended. The argument supplied to SEEK is an absolute number of bytes from the start of the file. If you are using random access files and want to access the 7th record in the file, then you need to multiply your record size by 7 to get the final location to seek to.

Associated

CLOSE, EOF, FFWD, INPUT#, OPEN, PRINT#, REWIND, SEEK

Example

```
handle = OPEN( "TestFile.txt" )
recsize = 20
FOR recno = 0 TO 10 LOOP
   record$ = "Record " + STR$( recno )
   pad = recsize - LEN( record$ )
   PRINT# handle, record$; SPACE$( pad )
REPEAT
REM read the 7th record
SEEK ( handle, ( recsize + 1 ) * 7 )
INPUT# handle, record$
PRINT record$
CLOSE( handle )
END
```

SENSEACCELX / Y / Z

Purpose

Returns the value of the Raspberry Pi senseHAT accelerometer.

Syntax

value = SENSEACCELX
value = SENSEACCELY
value = SENSEACCELZ

Description

The **Raspberry Pi senseHAT** has a number of built in sensors. The accelerometer can be accessed with this function.

Associated

SENSECOMPASS, SENSEGYRO, SENSEACCEL

Example

```
CLS
LOOP
PRINT "Sense Accelerometer X="; SENSEACCELX
PRINT "Sense Accelerometer Y="; SENSEACCELY
PRINT "Sense Accelerometer Z="; SENSEACCELZ
REPEAT
END
```

SENSECLS

Purpose

Sets all of the LEDs on the Raspberry Pi senseHAT to off.

Syntax

SENSECLS

Description

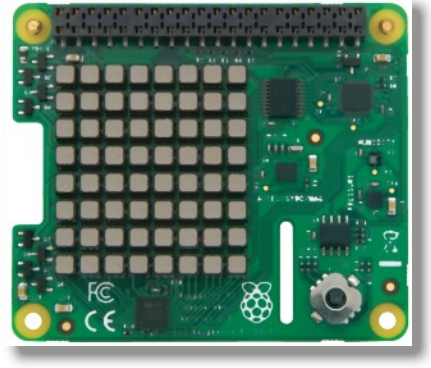
Sets an RGB value of 0, 0, 0 to all of the matrix LEDs thereby clearing the display.

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

Example

```
CLS
SENSEPLOT( 2, 2 )
WAIT ( 1 )
SENSECLS
END
```



SENSECOMPASSX / Y / Z

Purpose

Returns the value of the Raspberry Pi senseHAT compass.

Syntax

value = SENSECOMPASSX
value = SENSECOMPASSY
value = SENSECOMPASSZ

Description

The Raspberry Pi senseHAT has a number of built in sensors. The compass can be accessed with this function.

Associated

SENSECOMPASS, SENSEGYRO, SENSEACCEL

Example

```
CLS
LOOP
PRINT "Sense Compass X="; SENSECOMPASSX
PRINT "Sense Compass Y="; SENSECOMPASSY
PRINT "Sense Compass Z="; SENSECOMPASSZ
REPEAT
END
```

SENSEGETRGB

Purpose

Return the values used by a Raspberry Pi senseHAT LED

Syntax

SENSEGETRGB(*xpos, ypos, red, green, blue*)

Description

Returns the Red, Green and Blue values from a given LED at the specified matrix coordinates. It is possible to use this for collision detection in games using the senseHAT.

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

Example

```
CLS
SENSERGBCOLOUR( 0, 0, 255 )
SENSEPLOT( 0, 0 )
SENSEGETRGB( 0, 0, r, g, b )
PRINT "Red="; r
PRINT "Green="; g
PRINT "Blue="; b
END
```

SENSEGYROX / Y / Z

Purpose

Returns the value of the Raspberry Pi senseHAT gyro.

Syntax

value = SENSEGYROX
value = SENSEGYROY
value = SENSEGYROZ

Description

The Raspberry Pi senseHAT has a number of built in sensors. The gyro can be accessed with this function.

Associated

SENSECOMPASS, SENSEGYRO, SENSEACCEL

Example

```
CLS
LOOP
PRINT "Sense Gyro X="; SENSEGYROX
PRINT "Sense Gyro Y="; SENSEGYROY
PRINT "Sense Gyro Z="; SENSEGYROZ
REPEAT
END
```

SENSEHEIGHT

Purpose

Returns the value of the Raspberry Pi senseHAT height sensor.

Syntax

value = SENSEHEIGHT

Description

The Raspberry Pi senseHAT has a number of built in sensors. Height above sea level can be accessed with this function. Note, this does not work well indoors.

Associated

SENSEHEIGHT, SENSEHUMIDITY, SENSEPRESSURE, SENSETEMPERATURE

Example

```
CLS
LOOP
PRINT "Height"; SENSEHEIGHT
REPEAT
END
```

SENSEHFLIP / SENSEVFLIP

Purpose

Horizontally flips the Raspberry Pi senseHAT LED matrix.

Syntax

SENSEHFLIP
SENSEVFLIP

Description

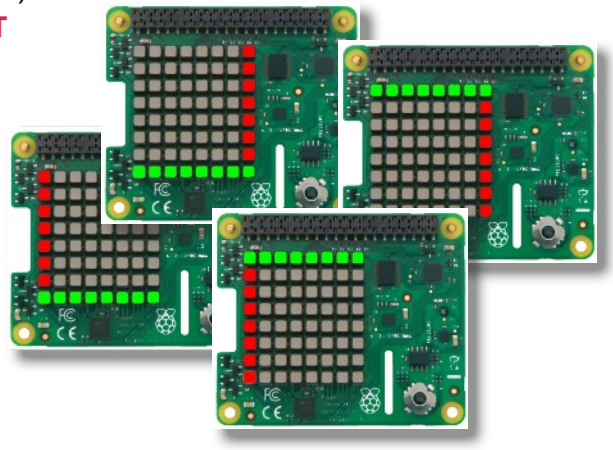
Reverses (flips) the LED matrix display horizontally or vertically.

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

Example

```
CLS
SENSERGBCOLOUR( 255, 0, 0 )
SENSELINE( 0, 0, 0, 7 )
SENDRGBCOLOUR( 0, 255, 0 )
SENSELINE( 0, 7, 7, 7 )
LOOP
SENSEHFLIP
WAIT( 1 )
SENSEVFLIP
WAIT( 1 )
REPEAT
END
```



SENSEHUMIDITY

Purpose

Returns the value of the Raspberry Pi senseHAT humidity sensor.

Syntax

value = SENSEHUMIDITY

Description

The **Raspberry Pi senseHAT** has a number of built in sensors. Humidity can be accessed with this function.

Associated

SENSEHEIGHT, SENSEHUMIDITY, SENSEPRESSURE, SENSETEMPERATURE

Example

```
CLS
LOOP
PRINT "Humidity"; SENSEHUMIDITY
REPEAT
END
```

SENSELINE

Purpose

Lights a line on the Raspberry Pi senseHAT LED matrix.

Syntax

SENSELINE(*x1, y1, x2, y2*)

Description

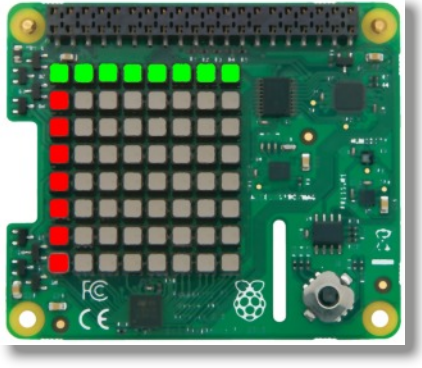
Sets the RGB values, defined by SENSERGBCOLOUR, to a line of LEDs on the Raspberry Pi senseHAT LED matrix. The line is displayed from x1, y1 to x2, y2.

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

Example

```
CLS
SENSERGBCOLOUR( 255, 0, 0 )
SENSELINE( 0, 0, 0, 7 )
SENSERGBCOLOUR( 0, 255, 0 )
SENSELINE( 0, 7, 7, 7 )
END
```



SENSEPLOT

Purpose

Lights an LED on the Raspberry Pi senseHAT LED matrix.

Syntax

SENSEPLOT(*xpos, ypos*)

Description

Sets the RGB values, defined by SENSERGBCOLOUR, to a single LED on the Raspberry Pi senseHAT LED matrix.

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

Example

```
CLS
SENSERGBCOLOUR( 255, 0, 0 )
SENSEPLOT( 0, 0 )
SENSERGBCOLOUR( 0, 255, 0 )
SENSEPLOT( 7, 7 )
END
```

SENSEPRESSURE

Purpose

Returns the value of the Raspberry Pi senseHAT air pressure sensor.

Syntax

value = SENSEPRESSURE

Description

The **Raspberry Pi senseHAT** has a number of built in sensors. Air pressure can be accessed with this function.

Associated

SENSEHEIGHT, SENSEHUMIDITY, SENSEPRESSURE, SENSETEMPERATURE

Example

```
CLS
LOOP
PRINT "Pressure"; SENSEPRESSURE
REPEAT
END
```

SENSERECT

Purpose

Lights a rectangle on the Raspberry Pi senseHAT LED matrix.

Syntax

SENSERECT(*xpos, ypos, width, height, fill*)

Description

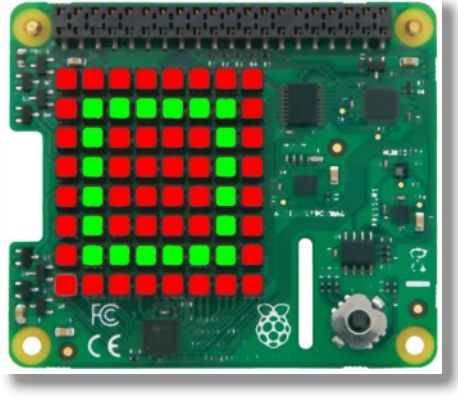
Sets the RGB values, defined by SENSERGBCOLOUR, to a rectangle of LEDs on the Raspberry Pi senseHAT LED matrix. The rectangle is displayed from xpos, ypos with a width and height as specified. Fill can be either 0 for an outline or 1 for filled in.

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

Example

```
CLS
SENSERGBCOLOUR( 255, 0, 0 )
SENSERECT( 0, 0, 7, 7, 1 )
SENSERGBCOLOUR( 0, 255, 0 )
SENSERECT( 1, 1, 5, 5, 0 )
END
```



SENSERGBCOLOUR

Purpose

Set the Red, Green and Blue values used by a Raspberry Pi senseHAT LED.

Syntax

SENSERGBCOLOUR(*red, green, blue*)

Description

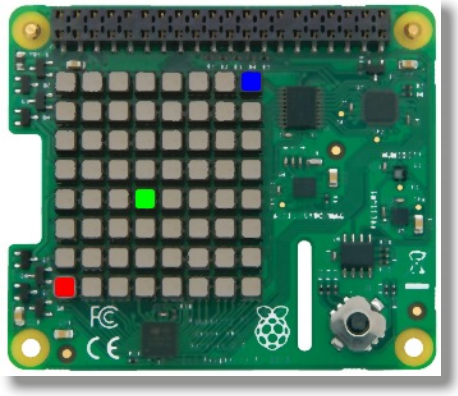
Sets the Red, Green and Blue values to be used by the senseHAT FUZE BASIC drawing commands.

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

Example

```
CLS
SENSERGBCOLOUR ( 255, 0, 0 )
SENSEPLOT ( 0, 0 )
SENSERGBCOLOUR ( 0, 255, 0 )
SENSEPLOT  ( 3, 3 )
SENSERGBCOLOUR ( 0, 0, 255 )
SENSEPLOT  ( 7, 7 )
END
```



SENSESCROLL

Purpose

Scrolls the Raspberry Pi senseHAT LED matrix.

Syntax

SENSESCROLL(*direction, direction*)

Description

Shifts the Raspberry Pi senseHAT LED matrix in the specified direction by the number of pixels indicated.

Example

```
SENSECLS
SENSEPLOT( 2, 2 )
SENSEPLOT( 2, 3 )
LOOP
SENSESCROLL ( 0, 2 ) // two up
WAIT(0.1)
SENSESCROLL ( 2, 0 ) // two right
WAIT(0.1)
SENSESCROLL ( 0, - 2 ) // two down
WAIT(0.1)
SENSESCROLL ( - 2, 0 ) // two left
WAIT(0.1)
REPEAT
END
```

Associated

SENSECLS, SENSEPLOT, SENSERECT, SENSESCROLL, SENSEHFLIP, SENSEVFLIP, SENSERGBCOLOUR, SENSEGETRGB, SENSELINE

SENSETEMPERATURE

Purpose

Returns the value of the Raspberry Pi senseHAT heat sensor.

Syntax

value = SENSETEMPERATURE

Description

The **Raspberry Pi senseHAT** has a number of built in sensors. Temperature, in degrees, can be accessed with this function.

Associated

SENSEHEIGHT, SENSEHUMIDITY, SENSEPRESSURE, SENSETEMPERATURE

Example

```
CLS
LOOP
PRINT "Temperature"; SENSETEMPERATURE
REPEAT
END
```

SETALPHA

Purpose

Set the level of transparency for graphics

Syntax

SETALPHA(*value*)

Description

Sets the level of transparency to be used by all drawing and text commands (0 - 255). 0 is totally transparent, 255 is opaque (solid).

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
image = loadImage ("img1.png")
setMode (getImageW (image), getImageH (image))
text$ = "FUZE BASIC"
alpha = 255
cx = GWIDTH / 2
cy = GHEIGHT / 2
cxdir = 2
cydir = 2
PaperOff
fontSize (6)
updateMode = 0
LOOP
 CLS2
 IF leftClick AND alpha > 0 THEN alpha = alpha - 1
 IF rightClick AND alpha < 255 THEN alpha = alpha + 1
 plotImage (image, 0, 0)
 setAlpha (alpha)
 COLOUR = YELLOW
 cx = cx + cxdir
 cy = cy + cydir
 IF cx < 0 OR cx > GWIDTH THEN cxdir =  - cxdir
 IF cy < 0 OR cy > GHEIGHT THEN cydir =  - cydir
 CIRCLE (cx, cy, 100, 1)
 INK = RASPBERRY
 printAt (TWIDTH / 2 - LEN (text$) / 2, THEIGHT / 2 - 1);  text$
 UPDATE
REPEAT
```

SETCHANVOL

Purpose

Set the volume of a sound sample.

Syntax

SETCHANVOL(*channel*, *volume*)

Description

Sets the sound sample playback volume on the specified *channel* where *volume* is a percentage of the maximum (0 - 100)

Example

```
channel = 0
ufo = LOADSAMPLE ("senemy.wav")
LOOP
  FOR volume = 20 TO 100 STEP 5 LOOP
    SETCHANVOL (channel, volume)
    PLAYSAMPLE (ufo, 0, 0)
    WAIT (0.1)
  REPEAT
REPEAT
END
```

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

SETDEVICE

Purpose

Checks which devices are connected (Arduino / compatible or BBC micro:bit)

Syntax

PRINT DEVICETYPE(ID)

Description

Returns a value corresponding to the device type:

1 = BBC micro:bit
2 = Arduino or compatible device

Note:

If you have more than one device attached then please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE

Associated

DETECTDEVICES, DEVICETYPE, SETDEVICE

Example

```
devices=DETECTDEVICES
CLS
IF devices = FALSE THEN
PRINT "No devices found"
END
ELSE
PRINT devices; " devices found"
PRINT
FOR id = 0 TO devices LOOP
IF DEVICETYPE(id) = 1 THEN
PRINT "BBC micro:bit :ID="; id
ENDIF
IF DEVICETYPE(id) = 2 THEN
PRINT "Arduino or compatible device :ID="; id
ENDIF
REPEAT
END
```

SETIMAGEORIGIN

Purpose

Set the point of origin for an image

Syntax

SETIMAGEORIGIN (handle, x offset, y offset)

Description

Allows you to set the drawing point of an image. This is useful when you need to centre images or draw an image using the mouse.

Associated

CLS, CLS2, COPYREGION, FREEIMAGE, FULLSCREEN, GETIMAGEH, GETIMAGEW, GRABREGION, LOADIMAGE, PLOTIMAGE, ROTATEIMAGE, SAVEREGION, SAVESCREEN, SCALEIMAGE, SCROLLDOWN, SCROLLLEFT, SCROLLRIGHT, SCROLLUP, SETIMAGEORIGIN, SETMODE, UPDATE, UPDATEREGION

Example

```
image = LOADIMAGE ("img1.png")
SETMODE (1280, 720)
x = GETIMAGEW (image) / 2
y = GETIMAGEH (image) / 2
PAPEROFF
FONTSIZE (6)
UPDATEMODE = 0
LOOP
 CLS2
 IF LEFTCLICK THEN SETIMAGEORIGIN (image, 0, 0)
 IF RIGHTCLICK THEN
 x = GETIMAGEW (image) / 2
 y = GETIMAGEH (image) / 2
 SETIMAGEORIGIN (image, x, y)
 ENDIF
 PLOTIMAGE (image, GWIDTH / 2, GHEIGHT / 2)
 COLOUR = RND (30)
 CIRCLE (GWIDTH / 2, GHEIGHT / 2, 30, 0)
 UPDATE
REPEAT
```

SETMODE

Purpose

Set display width and height

Syntax

SETMODE(*width*, *height*)

Description

Sets the display width and height to the specified. It is generally sensible to use standard screen display sizes.

Associated

FULLSCREEN, GHEIGHT, GWIDTH, ORIGIN, SETMODE

Example

```
SETMODE( 1280, 720 )
PRINT "Hello World"
WAIT( 2 )
SETMODE( 640, 480 )
PRINT "Hello Another World"
END
```

SETMOUSE

Purpose

Move the mouse pointer to the specified point.

Syntax

SETMOUSE(*xpos*, *ypos*)

Description

Moves the mouse pointer to the screen coordinate (*xpos*,*ypos*)

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
CLS
LOCKMOUSE(TRUE)
MOUSEON
COLOUR = WHITE
RECT (100, 100, 150, 50, TRUE)
INK = BLACK
PAPER = WHITE
PLOTTEXT ("Click Me", 105, 125);
UPDATE
clicked = FALSE
LOOP
 GETMOUSE (x, y, z)
 IF z <> 0 THEN
  IF (x > 100 AND x < 250) THEN
   IF (y > 100 AND y < 150) THEN
    clicked = TRUE
   ENDIF
  ENDIF
 ENDIF
REPEAT UNTIL clicked
SETMOUSE (GWIDTH / 2, GHEIGHT / 2)
LOCKMOUSE(FALSE)
END
```

SETMUSICVOL

Purpose

Sets the music playback volume.

Syntax

SETMUSICVOL(*level)*

Description

Sets the music playback volume where *level* is a percentage of the maximum (0-100)

Example

```
handle = LOADMUSIC( "forNext.mp3" )
SETMUSICVOL( 70 )
PLAYMUSIC( handle, 1 )
```

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

SETSPRITEALPHA

Purpose

Sets the transparency of a sprite

Syntax

SETSPRITEALPHA(*sprite*, *alpha*)

Description

Sets how transparent a sprite is. An alpha of 0 means it's invisible and 255 means it's completely opaque, or solid.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SAVESPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
sprite = NEWSPRITE (1)
image = LOADIMAGE ("img.png")
imagex = GWIDTH / 2 - GETIMAGEW (image) / 2
PLOTIMAGE (image, imagex, 0)
LOADSPRITE ("logo.png", sprite, 0)
spritex = GETSPRITEW (sprite) / 2
spritey = GETSPRITEH (sprite) / 2
SETSPRITEORIGIN (sprite, spritex, spritey)
FOR alpha = 0 TO 255 LOOP
  SETSPRITEALPHA( sprite, alpha )
  PLOTSPRITE (sprite, GWIDTH / 2, GHEIGHT / 2, 0)
  UPDATE
  WAIT (0.01)
REPEAT
END
```

SETSPRITEANGLE

Purpose

Rotate a sprite to the given angle

Syntax

SETSPRITEANGLE(*sprite*, *angle*)

Description

Use to rotate the specified sprite to the given angle in degrees. 0 is default.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
sprite = NEWSPRITE( 1 )
LOADSPRITE( "logo.png", sprite, 0 )
spritex = GETSPRITEW (sprite) / 2
spritey = GETSPRITEH (sprite) / 2
SETSPRITEORIGIN (sprite, spritex, spritey)
LOOP
FOR angle = 0 TO 360 STEP 0.01 LOOP
SETSPRITEANGLE( sprite, angle )
PLOTSPRITE( sprite, GWIDTH / 2, GHEIGHT / 2, 0 )
UPDATE
REPEAT
REPEAT
END
```

SETSPRITEFLIP

Purpose

Mirror a sprite in the specified direction

Syntax

SETSPRITEFLIP(*sprite*, *flip*)

Description

Graphically mirrors (flips) the specified sprite.

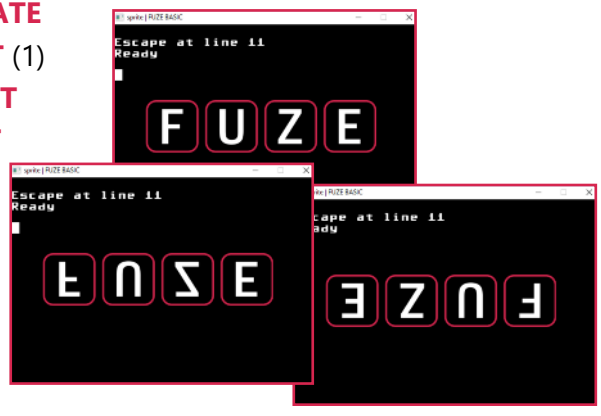
0 Reset to default
1 mirrored vertically
2 mirrored horizontally
3 mirrored vertically & horizontally

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
sprite = NEWSPRITE (1)
LOADSPRITE ("logo.png", sprite, 0)
spritex = GETSPRITEW (sprite) / 2
spritey = GETSPRITEH (sprite) / 2
SETSPRITEORIGIN (sprite, spritex, spritey)
PLOTSPRITE (sprite, GWIDTH / 2, GHEIGHT / 2, 0)
LOOP
  FOR a = 3 TO 0 STEP -1 LOOP
    SETSPRITEFLIP (sprite, a)
    UPDATE
    WAIT (1)
  REPEAT
REPEAT
```



SETSPRITEORIGIN

Purpose

Sets the anchor point of a sprite

Syntax

SETSPRITEorigin(*sprite, xpos, ypos*)

Description

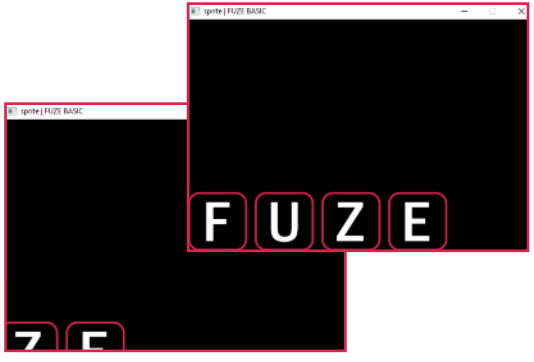
Use to set the origin of a specified sprite. When plotting a sprite its default origin is bottom left. You can change this to any point on the sprite. This example sets it to the middle.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
sprite = newSprite (1)
loadSprite ("logo.png", sprite, 0)
LOOP
  setSpriteOrigin (sprite, 0, 0)
  plotSprite (sprite, 0, 0, 0)
  UPDATE
  WAIT (.5)
  middlex = getSpriteW (sprite) / 2
  middley = getSpriteH (sprite) / 2
  setSpriteOrigin (sprite, middlex, middley)
  plotSprite (sprite, 0, 0, 0)
  UPDATE
  WAIT (.5)
REPEAT
```



SETSPRITESIZE

Purpose

Change the size of a sprite

Syntax

SETSPRITESIZE(*sprite*, *size*)

Description

Sets the sprite to the specified *size* in percent. 100 is the default, therefore 50 is half the size and 300 is three times as big as the original.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
sprite = NEWSPRITE( 1 )
LOADSPRITE( "logo.png", sprite, 0 )
FOR size = 1 TO 300 step .1 LOOP
SETSPRITESIZE( sprite, size )
x=(GWIDTH / 2)-GETSPRITEW (sprite) / 2
y=(GHEIGHT / 2)-GETSPRITEH (sprite) / 2
PLOTSPRITE( sprite, x, y, 0 )
UPDATE
REPEAT
END
```

SGET

Purpose

Read a byte from a serial port.

Syntax

byte = SGET(*handle*)

Description

Fetch a single byte of data from an open serial port and return the data as a number. This function will pause program execution for up to 5 seconds if no data is available. If there is still not data after 5 seconds, the function will return -1.

Associated

SCLOSE, SGET, SGET$, SOPEN, SPUT, SPUT$, SREADY

Example

```
REM Read a byte from a serial port
arduino = SOPEN("/dev/ttyUSB0", 115200 )
byte = SGET( arduino )
PRINT byte
SCLOSE( arduino )
END
```

SGET$

Purpose

Read a character from a serial port.

Syntax

character = SGET$(*handle*)

Description

Fetch a single byte of data from an open serial port and return the data as a single character string. This function will pause program execution for up to 5 seconds if no data is available. If there is still not data after 5 seconds, the function will return an empty string.

Associated

SCLOSE, SGET, SGET$, SOPEN, SPUT, SPUT$, SREADY

Example

```
REM Read a character from a serial port
arduino = SOPEN( "/dev/ttyUSB0", 115200 )
char$ = SGET$( arduino )
PRINT char$
SCLOSE( arduino )
END
```

SGN

Purpose

Returns the sign of the specified number.

Syntax

sign = SGN(*number*)

Description

Returns -1 if the number is negative, 1 otherwise. (Zero is considered positive)

Associated

ABS, SGN

Example

```
REM Prints 1
PRINT SGN( 100 )
PRINT SGN( 0 )
REM Print -1
PRINT SGN( -5 )
END
```

SIN

Purpose

Returns the sine of the given angle.

Syntax

sine = SIN(*angle*)

Description

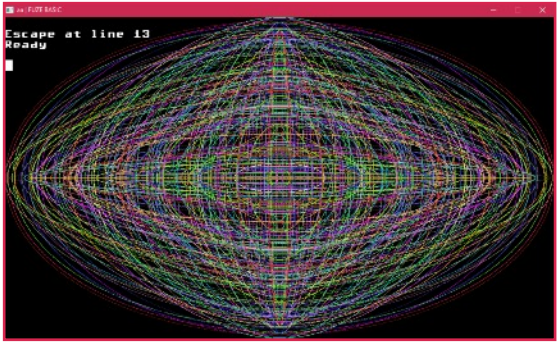
Returns the sine of the argument *angle* in radians. This is the ratio of the side of a right angled triangle, that is opposite to the angle, to the hypotenuse (the longest side).

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
REM Draw an ellipse in the screen centre
CLS
RAD
LOOP
width=RND(GWIDTH/2)
height=RND(GHEIGHT/2)
RGBCOLOUR (RND(255), RND(255), RND(255))
FOR angle = 0 TO 6 * PI STEP 0.01 LOOP
xpos = width * COS( angle ) + GWIDTH / 2
ypos = height * SIN( angle ) + GHEIGHT / 2
PLOT( xpos, ypos )
REPEAT
UPDATE
REPEAT
END
```



SOFTPWMWRITE

Purpose

Synthesize Pulse Wave Modulation to the specified GPIO pin.

Syntax

SOFTPWMWRITE(*pinNo*, *value*)

Description

This enables you to simulate an analog output. For example instead of an LED being just on or off you can make it appear brighter or dimmer. The *pinNo* parameter is the number of the GPIO output pin. This must first be set to *pinSoftPwm*. The *value* parameter is a percentage (0-100).

NOTE: For FUZE and RPi only

Associated

ANALOGREAD, ANALOGWRITE, DIGITALREAD, DIGITALWRITE, PINMODE, PWMWRITE, SOFTPWMWRITE

Example

```
REM Connect an LED to GPIO pin 0
PINMODE( 0, PINSOFTPWM )
FOR i = 0 TO 100 LOOP
  SOFTPWMWRITE ( 0, i )
  WAIT( 0.1 )
REPEAT
FOR i = 100 to 0 STEP -1 LOOP
  SOFTPWMWRITE ( 0, i )
  WAIT( 0.1 )
REPEAT
END
```

SOPEN

Purpose

Opens a serial device and makes it available for use.

Syntax

handle = SOPEN(*device*, *speed*)

Description

This opens a serial device and makes it available for our use. It takes the name of the serial port and the speed as an argument and returns a number (the handle) of the device. We can use this handle to reference the device and allow us to open several devices at once. The following baud rates are recognised: 50, 75, 110, 134, 150,200, 300, 600, 1200, 1800, 2400, 19200, 38400, 57600, 115200 and 230400, but do check your local PC and devices capabilities. The device is always opened with the data format set to 8 data bits, 1 stop bit and no parity. All handshaking is turned off.

Associated

SCLOSE, SGET, SGET$, SOPEN, SPUT, SPUT$, SREADY

Example

```
REM Read a byte from a serial port
arduino = SOPEN( "/dev/ttyUSB0", 115200 )
byte = SGET( arduino )
SCLOSE( arduino )
END
```

SPACE$

Purpose

Returns a blank string of the specified length.

Syntax

*blank$* = SPACE$(*number*)

Description

Returns a string of blank spaces *number* characters long.

Example

```
CLS
blank$ = SPACE$( 30 )
FOR i = 1 TO 20 LOOP
PRINT i; blank$; i
REPEAT
END
```

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

SPLINE

Purpose

Interpolate a smooth curve between two points

Syntax

value = SPLINE (start, point1, point2, end, controlPoint)

Description

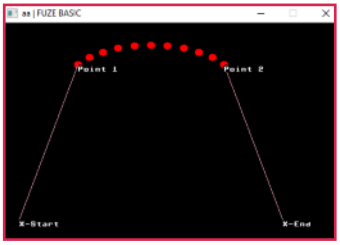
Creates a smooth curve passing through or near to, a specified number of points. 'start' indicates where the curve has come from; points 1 and 2 are the points the curve will pass through (or near depending on the start and end points) and 'end' is where the curve would end.

'controlPoint' is the distance along the curve specified by a number between 0 (the beginning) and 1 (the end). Therefore to return the exact middle of the curve a control point of 0.5 would be used.

Associated

LERP

Example




```
FONTSIZE (1)
xpoint1 = (GWIDTH / 10) * 4
xpoint2 = (GWIDTH / 10) * 6
top = (GHEIGHT / 10) * 8
LOOP
 CLS2
 FOR sp = 0 TO 1 STEP 0.1 LOOP
  bottom = MOUSEY
  xStart = GWIDTH / 2 - ABS (GWIDTH / 2 - MOUSEX)
  xEnd = GWIDTH / 2 + ABS (GWIDTH / 2 - MOUSEX)
  x = SPLINE (xStart, xpoint1, xpoint2, xEnd, sp)
  y = SPLINE (bottom, top, top, bottom, sp)
  COLOUR = RED
  CIRCLE (x, y, 5, 1)
  COLOUR = LIGHTPINK
  LINE (xStart, bottom, xpoint1, top)
  LINE (xEnd, bottom, xpoint2, top)
  PLOTTEXT ("X-Start", xStart, bottom)
  PLOTTEXT ("Point 1", xpoint1, top)
  PLOTTEXT ("Point 2", xpoint2, top)
  PLOTTEXT ("X-End", xEnd, bottom)
 REPEAT
 UPDATE
REPEAT
```

SPRITECENTRE

Purpose

Sets the anchor point of a sprite to its centre

Syntax

SPRITECENTRE(*sprite*)

Description

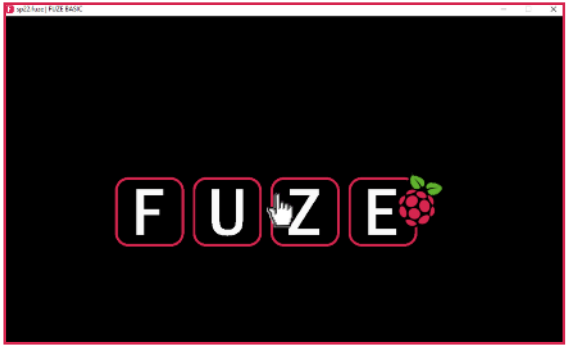
Use to set the origin of a specified sprite to its exact centre.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
logo = NEWSPRITE ( 1 )
hand = NEWSPRITE ( 1 )
LOADSPRITE ("hand1.png", hand, 0 )
SPRITECENTRE ( hand )
LOADSPRITE ("fuzelogo.png", logo, 0 )
LOOP
 PLOTSPRITE ( logo, MOUSEX, MOUSEY, 0 )
 PLOTSPRITE ( hand, MOUSEX, MOUSEY, 0 )
 IF LEFTCLICK THEN
  SPRITECENTRE ( logo )
 ELSE
  IF RIGHTCLICK THEN
   SETSPRITEORIGIN ( logo, 0, 0 )
  ENDIF
 ENDIF
 UPDATE
REPEAT
```



SPRITECOLLIDE

Purpose

Detect a sprite collision (fast bounding box)

Syntax

collision = SPRITECOLLIDE(*target*)

Description

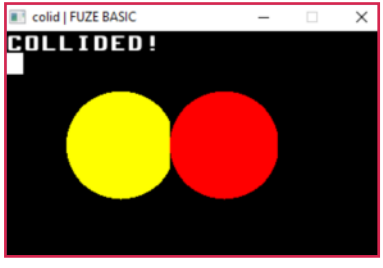
Returns the sprite index of the first sprite that the sprite index *target* has collided with, or -1 if there is no collision. It only checks the current sprite location and this is only updated after a screen update LOOP so it is possible to call PLOTSPRITE() and have a sprite overlap but it not be detected until after the update has happened on screen.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
COLOUR = YELLOW
CIRCLE (100, 100, 50, TRUE)
SAVEREGION ("s1.bmp", 50, 50, 100, 100)
COLOUR = Red
CIRCLE (100, 100, 50, TRUE)
SAVEREGION ("s2.bmp", 50, 50, 100, 100)
CLS2
s1 = NEWSPRITE (1)
s2 = NEWSPRITE (1)
LOADSPRITE ("s1.bmp", s1, 0)
LOADSPRITE ("s2.bmp", s2, 0)
FOR X = 0 TO GWIDTH STEP 1 LOOP
  PLOTSPRITE (s1, X, 200, 0)
  PLOTSPRITE (s2, GWIDTH - X - 100, 200, 0)
  IF SPRITECOLLIDE (s2) <> -1 THEN
    PRINT "COLLIDED!"
    END
  ENDIF
  UPDATE
  WAIT (0.01)
REPEAT
END
```



SPRITECOLLIDEPP

Purpose

Detect a sprite collision (pixel perfect)

Syntax

collision = SPRITECOLLIDEPP(*target*, *accuracy*)

Description

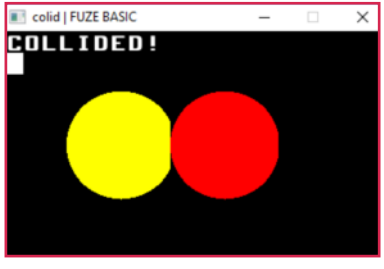
This is a slower but more accurate version of SPRITECOLLIDE. It first does do a simple bounding box test then checks row at a time. The *accuracy* parameter is how many pixels to skip both horizontally and vertically. This is from 1 to 16, where 1 is "perfect" and greater than 1 is less accurate, but faster. This will affect the amount of visible overlap you get before a collision is detected.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
CLS
COLOUR = YELLOW
CIRCLE( 100, 100, 50, TRUE )
SAVEREGION( "s1.bmp", 50, 50, 101, 101 )
COLOUR = RED
CIRCLE( 100, 100, 50, TRUE )
SAVEREGION( "s2.bmp", 50, 50, 101, 101 )
CLS2
s1 = NEWSPRITE( 1 )
s2 = NEWSPRITE( 1 )
LOADSPRITE( "s1.bmp", s1, 0 )
LOADSPRITE( "s2.bmp", s2, 0 )
FOR x = 0 TO GWIDTH LOOP
PLOTSPRITE( s1, x, 200, 0 )
PLOTSPRITE( s2, GWIDTH - x - 100, 200, 0 )
 IF SPRITECOLLIDEPP (s2, 8) <> -1 THEN
  PRINT "COLLIDED!"
  END
 ENDIF
 UPDATE
 WAIT (0.01)
REPEAT
END
```



SPRITEOUT

Purpose

Find out if a sprite is off screen

Syntax

result = SPRITEOUT(*sprite*)

Description

If the sprite is off screen then result is true otherwise it is false.

Associated

ADVANCESPRITE, CLONESPRITE, GETSPRITEANGLE, GETSPRITEH, GETSPRITEW, GETSPRITEX, GETSPRITEY, HIDESPRITE, LOADSPRITE, NEWSPRITE, PLOTSPRITE, SETSPRITEALPHA, SETSPRITEANGLE, SETSPRITEFLIP, SETSPRITEORIGIN, SETSPRITESIZE, SPRITECOLLIDE, SPRITECOLLIDEPP, SPRITEOUT

Example

```
sprite = NEWSPRITE (1)
LOADSPRITE ("fighter.png", sprite, 0)
PLOTSPRITE (sprite, GWIDTH / 2, GHEIGHT / 2, 0)
LOOP
  SETSPRITEANGLE (sprite, RND (360))
  PLOTSPRITE (sprite, GWIDTH / 2, GHEIGHT / 2, 0)
  WHILE  NOT SPRITEOUT (sprite) LOOP
    ADVANCESPRITE (sprite, 1)
  REPEAT
REPEAT
END
```

SPUT

Purpose

Send a byte to an open serial port.

Syntax

SPUT(*arduino*,*byte*)

Description

Send a single byte of data to an open serial port.

Associated

SCLOSE, SGET, SGET$, SOPEN, SPUT, SPUT$, SREADY

Example

```
REM Write a byte to a serial port
arduino = SOPEN( "/dev/ttyUSB0", 115200 )
SPUT( arduino, 52 )
SPUT( arduino, 50 )
SCLOSE( arduino )
END
```

SPUT$

Purpose

Send a character string to an open serial port.

Syntax

SPUT$(*arduino*, *string*)

Description

Send a string of characters of data to an open serial port.

Associated

SCLOSE, SGET, SGET$, SOPEN, SPUT, SPUT$, SREADY

Example

```
REM Write a byte to a serial port
arduino = SOPEN( "/dev/ttyUSB0", 115200 )
SPUT$( arduino, "Hello" )
SCLOSE( arduino )
END
```

SQRT

Purpose

Return the square root of the specified number.

Syntax

squareroot = SQRT(*number*)

Description

Returns the square root of the argument *number*. This is the opposite of multiplying a number by itself i.e. X = SQRT(X * X).

Example

```
CLS
FONTSIZE (3)
INK= GREEN
PRINT "foursquared = 4 * 4"
foursquared = 4 * 4
PRINT
INK=LIGHTPINK
PRINT "The Squareroot of foursquared is ";SQRT( foursquared )
END
```

SREADY

Purpose

Get the number of characters available to be read on an open serial port.

Syntax

count = SREADY(*handle*)

Description

Returns the number of characters available to be read from an open serial port. This can be used to poll the device to avoid stalling your program when there is no data available to be read.

Associated

SCLOSE, SGET, SGET$, SOPEN, SPUT, SPUT$, SREADY

Example

```
REM Read a character from a serial port
arduino = SOPEN( "/dev/ttyUSB0", 115200 )
IF SREADY( arduino ) THEN
   char$ = SGET$( arduino )
   PRINT char$
ENDIF
SCLOSE( arduino )
END
```

STOP

Purpose

Stop a running program.

Syntax

STOP

Description

Program execution is stopped with a message indicating the current line number.

Associated

CONT

Example

```
CLS
FONTSIZE (3)
INK = WHITE
INPUT "Enter the Password: ", pass$
IF pass$ <> "wibble" THEN
  INK = RED
  PRINT "Password Incorrect"
  END
ENDIF
INK = GREEN
PRINT "Password Correct"
END
```

STOPCHAN

Purpose

Stop the playing of a sound sample.

Syntax

STOPCHAN(*handle*)

Description

This function stops the playing of the sound sample associated with the *handle* returned by *LOADSAMPLE* that has been started using *PLAYSAMPLE*. It cannot be resumed once stopped.

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

Example

```
channel = 0
volume = 70
SETCHANVOL( channel, volume )
intro = LOADSAMPLE( "senemy.wav" )
PLAYSAMPLE( intro, channel, 999 )
WAIT( 1 )
STOPCHAN( intro )
END
```

STOPMUSIC

Purpose

Stop music playing completely.

Syntax

STOPMUSIC

Description

Stops a playing music track which cannot then be resumed.

Example

```
channel = 0
volume = 70
SETCHANVOL( channel, volume )
intro = LOADMUSIC( "neon.mp3" )
PLAYMUSIC( intro, channel )
WAIT( 1 )
STOPMUSIC
END
```

Associated

LOADMUSIC, LOADSAMPLE, PAUSEMUSIC, PLAYMUSIC, PLAYSAMPLE, RESUMECHAN, RESUMEMUSIC, SETMUSICVOL, STOPCHAN, STOPMUSIC

STR$

Purpose

Returns a string version of the supplied number.

Syntax

*string$* = STR$(*number*)

Description

Returns a string in the decimal (base 10) representation of *number*. This is useful if you want to append a number to a string. This is the opposite of the VAL function.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
PRINT "The Answer is " + STR$( 42 )
END
```

SWAP

Purpose

Swap the value of two variables.

Syntax

SWAP(*value1*, *value2*)

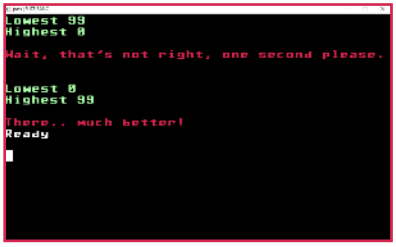
Description

This swaps the value of the 2 variables round. Both arguments must be the same type - i.e. Both numeric or both string.

Associated

MIN, MAX

Example




```
CLS
FONTSIZE(3)
lowest = 99
highest = 0
INK=LIGHTGREEN
PRINT "Lowest "; lowest
PRINT "Highest "; highest
WAIT (1)
PRINT
INK=RASPBERRY
PRINT "Wait, that's not right, one second please."
WAIT (1.5)
IF lowest > highest THEN SWAP( highest, lowest )
INK = LIGHTGREEN
PRINT
PRINT "Lowest "; lowest
PRINT "Highest "; highest
WAIT(1)
INK=RASPBERRY
PRINT
PRINT "There.. much better!"
END
```

SWITCH, CASE, DEFAULT, ENDCASE, ENDSWITCH

Purpose

Test a value against many different values and execute different code.

Syntax

SWITCH (*variable*)
  { CASE *value* {, *value* }
    commands
  ENDCASE }
  [DEFAULT
    commands
  ENDCASE]
ENDSWITCH

Description

Simplify the writing of multiple IF... THEN... ELSE statements.

Every SWITCH must have a matching ENDSWITCH.

Every CASE or DEFAULT statement must have a matching ENDCASE.

Statements after a CASE statement must not run-into another CASE.

The constants after the CASE statement (and the expression in the SWITCH statement) can be either numbers or strings, but you can't mix both.

Associated

CASE, DEFAULT, ELSE, ENDIF, ENDCASE, ENDSWITCH, IF THEN, SWITCH

Example

```
INPUT a
SWITCH( a )
   CASE 1, 2
      PRINT "You entered 1 or 2"
   ENDCASE
   CASE 7
      PRINT "You entered 7"
   ENDCASE
   DEFAULT
      PRINT "You entered something else"
   ENDCASE
ENDSWITCH
END
```

TAN

Purpose

Return the tangent of the given angle.

Syntax

tangent = TAN(*angle*)

Description

Returns the tangent of the *angle* in the current angle units. In a right-angled triangle the tangent of an angle is the ratio of the length of the opposite side to the length of the adjacent side. This is a measure of the steepness of an angle.

Associated

ACOS, ASIN, ATAN, COS, CLOCK, DEG, PI, PI2, RAD, SIN, TAN

Example

```
DEG
PRINT "Tangent of 45 degrees: "; TAN( 45 )
PRINT "ArcTangent of 1: "; ATAN( 1 )
END
```

TANGLE

Purpose

Read or set the current angle of the turtle.

Syntax

angle = TANGLE
TANGLE = *angle*

Description

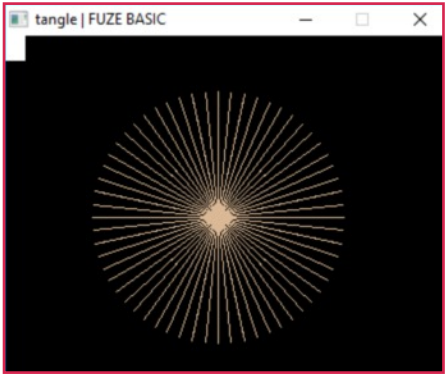
This can be read or assigned to and represents the current angle of the turtle when using turtle graphics mode (in the current angle units)

Associated

LEFT, MOVE, MOVETO, PENDOWN, PENUP, RIGHT, TANGLE

Example

```
CLS
ORIGIN (GWIDTH / 2, GHEIGHT / 2)
CLOCK
COLOUR = LIGHTBROWN
FOR i = 1 TO 60 LOOP
  TANGLE = i
  MOVETO (0, 0)
  PENDOWN
  MOVE (100)
REPEAT
END
```



THEIGHT / TWIDTH

Purpose

The height and width in characters of the display.

Syntax

height = THEIGHT
width = TWIDTH

Description

Returns the text character height and width of the display.

Associated

HTAB, HVTAB, INK, FONTSIZE, NUMFORMAT, PAPER, PAPEROFF, PAPERON, PLOTTEXT, PRINTAT, PRINT, TWIDTH, THEIGHT, VTAB

Example

```
CLS
text$ = "This text is centred in the screen"
PRINTAT (( TWIDTH - LEN( text$ )) / 2, THEIGHT / 2 );  text$
END
```

TIME

Purpose

Find out how long the program has been running in 1000s of a second.

Syntax

time = TIME

Description

This returns a number which represents the time that your program has been running in thousandths of a second.

Associated

MICROTIME

Example

```
REM Simple reaction timer
WAIT( 2 )
REM Make sure no key pressed
WHILE INKEY <> -1 LOOP
REPEAT
starttime = TIME
PRINT "Go!"
WHILE INKEY = -1 LOOP
REPEAT
endtime = TIME
PRINT "Your reaction time is ";
PRINT endtime -starttime ; " thousandths of a second"
END
```

TIME$

Purpose

Returns a string with the current time.

Syntax

*now$* = TIME$

Description

This returns a string with the current time in the following format: HH:MM:SS. For example: 18:05:45.

Example

```
PRINT "The time now is ";
PRINT TIME$
END
```

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

TONE

Purpose

Play a tone with the specified parameters.

Syntax

TONE(*channel*, *volume*, *frequency*, *duration*)

Description

This plays a simple tone of the given *frequency* (1 to 5000Hz), *volume* (%) and *duration* (0.01 to 20 seconds) on the given *channel*.

You can play multiple tones by playing them one after the other and up to three tones can be played simultaneously on different channels.

Channel 0 is white noise.

Example

```
channel = 1
volume = 70
FOR note = 1 TO 5 LOOP
   READ frequency, duration
   TONE( channel, volume, frequency, duration )
REPEAT
END
DATA 440, 1, 493, 1, 392, 1, 196, 1, 294 ,2
```

TRIANGLE

Purpose

Draw a triangle on the screen.

Syntax

TRIANGLE (*xpos1*, *ypos1*, *xpos2*, *ypos2*, *xpos3*, *ypos3*, *fill*)

Description

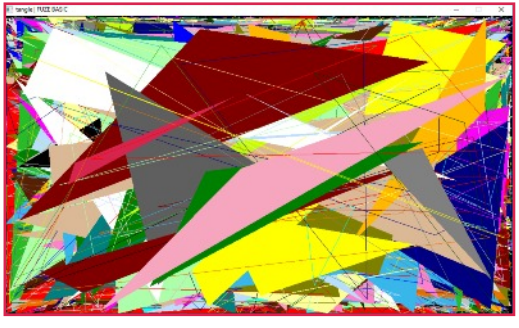
Draws a triangle with its corners at the three given points. The final parameter, *fill* is either **TRUE** or **FALSE**, and specifies filled (**TRUE**) or outline (**FALSE**).

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
LOOP
  COLOUR = RND( 30 )
  x1 = RND( GWIDTH )
  x2 = RND(GWIDTH )
  x3 = RND(GWIDTH )
  y1 = RND(GHEIGHT )
  y2 = RND(GHEIGHT )
  y3 = RND(GHEIGHT )
  f = RND( 2 )
  TRIANGLE ( x1, y1, x2, y2, x3, y3, f )
  UPDATE
  IF INKEY <> -1 THEN BREAK
REPEAT
END
```



TRUE

Purpose

Represents the logical "true" value.

Syntax

TRUE

Description

Represents a Boolean value that succeeds a conditional test. It is equivalent to a numeric value of 1 (in fact anything other than 0 evaluates to **TRUE**)

Associated

FALSE

Example

```
condition = TRUE
IF condition = TRUE THEN
   PRINT "Condition is TRUE"
ENDIF
PRINT condition
END
```

UPDATEMODE

Purpose

Set the video update mode.

Syntax

UPDATEMODE = *mode*

Description

UPDATEMODE determines when the screen is redrawn. Redrawing the screen takes a little time and will slow down a program if you do it too often. The value of the *mode* parameter can be **0**, **1**, or **2** as follows:

0- automatic updates do not happen. Nothing will be drawn on the screen until UPDATE is issued.

1-This is the default mode whereby an UPDATE happens automatically when you output a new line, or the screen scrolls.

2-The screen is updated after every PRINT instruction whether it takes a new line or not.

Associated

CLS, CLS2, UPDATE, UPDATEMODE, UPDATEREGION

Example

```
CLS
INPUT "Update Mode? ", mode
IF mode >= 0 AND mode <= 2 THEN
   PRINT "Press space to exit"
   WAIT( 1 )
   UPDATEMODE = mode
   LOOP
      PRINT "Hello World ";
   REPEAT UNTIL INKEY = 32
   UPDATE
ELSE
   PRINT "Invalid Update Mode"
ENDIF
WAIT( 1 )
END
```

UNTIL REPEAT

Purpose

Loop until the specified condition is met.

Syntax

UNTIL *condition* LOOP
  {*statements*}
REPEAT

Description

Execute the *statements* zero or more times until the *condition* is TRUE (Not 0).

Because the test is done at the start of the loop the *statements* may not be executed at all

Example

```
REM Print 1 to 10
count = 1
UNTIL count > 10 LOOP
  PRINT count
  count = count + 1
REPEAT
END
```

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT UNTIL, UNTIL REPEAT, WHILE REPEAT

UPDATE

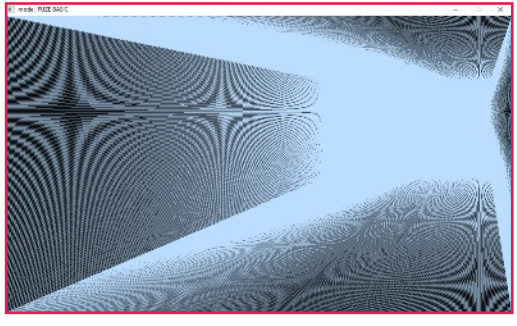
Purpose
Update screen graphics.

Syntax
UPDATE

Description
Graphics are drawn to a temporary screen buffer rather than the visible screen. The UPDATE command copies the working area to the main display. An update is also performed if your program stops for input, or when you PRINT a new line.

Associated
CLS, CLS2, UPDATE, UPDATEMODE, UPDATEREGION

Example
```
REM Moire patterns
LOOP
  CLS
  COLOUR = RND( 30 ) + 1
  x = RND( GWIDTH )
  y = RND( GHEIGHT )
  FOR w = 0 TO GWIDTH - 1 STEP 3 LOOP
    LINE( x, y, w, 0 )
    LINE( x, y, w, GHEIGHT - 1 )
  REPEAT
  FOR h = 0 TO GHEIGHT-1 STEP 3 LOOP
    LINE( x, y, 0, h )
    LINE( x, y, GWIDTH - 1, h )
  REPEAT
  UPDATE
REPEAT
END
```



UPDATEREGION

Purpose
Update a region of the screen.

Syntax
UPDATEREGION (x, y, width, height)

Description
Restricts the screen update to a specified rectangular area. This can really aid performance especially with games programming. If only a portion of your screen contains moving graphics then updating this are only will increase performance.

Associated
CLS, CLS2, UPDATE, UPDATEMODE, UPDATEREGION

Example
```
CLS
LOCKMOUSE (TRUE)
boxsize = 50
LOOP
rx = MOUSEX
ry = MOUSEY
IF LEFTCLICK AND boxsize > 10 THEN
boxsize = boxsize - 1
ENDIF
IF RIGHTCLICK AND boxsize < gHeight - MOUSEY THEN
boxsize = boxsize + 1
ENDIF
IF MOUSEX > GWIDTH - boxsize THEN rx = GWIDTH - boxsize
IF MOUSEY > GHEIGHT - boxsize THEN ry = GHEIGHT - boxsize
IF MIDDLECLICK THEN UPDATE
COLOUR = RND (30)
CIRCLE (RND (GWIDTH), RND (GHEIGHT), RND (100), 1)
COLOUR = WHITE
RECT (rx, ry, boxsize, boxsize, 0)
UPDATEREGION (rx, ry, boxsize, boxsize)
REPEAT
```

VAL

Purpose

Returns the number represented by a character string.

Syntax

number = VAL(*string$*)

Description

Returns the *number* represented by *string$*. This is the opposite of the STR$ function.

Associated

ASC, CHR$, DATE$, DEFCHAR, GET$, LEFT$, LEN, MID$, NUMFORMAT, PRINT, PRINTAT, RIGHT$, SPACE$, STR$, TIME$, VAL

Example

```
now$ = TIME$
hh = VAL( LEFT$( now$, 2 ))
mm = VAL( MID$( now$, 3, 2 ))
ss = VAL( RIGHT$( now$, 2 ))
elapsed = hh * 3600 + mm * 60 + ss
PRINT "Seconds since midnight: "; elapsed
END
```

VLINE

Purpose

Draws a vertical line.

Syntax

VLINE (*ypos1*, *ypos2*, *xpos*)

Description

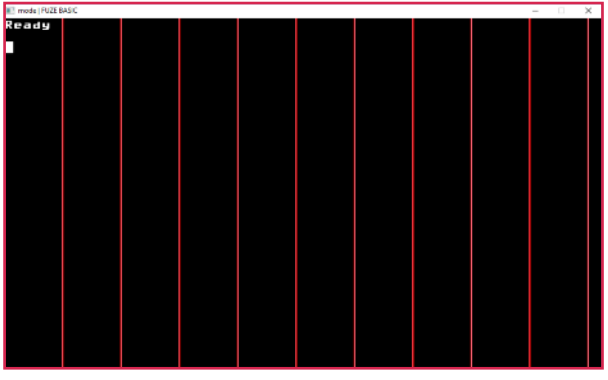
Draws a vertical line on column *xpos*, from row *ypos1* to row *ypos2*.

Associated

CIRCLE, COLOUR, ELLIPSE, HLINE, LINE, LINETO, PLOT, PLOTTEXT, POLYEND, POLYPLOT, POLYSTART, RGBCOLOUR, RECT, SETALPHA, TRIANGLE, VLINE

Example

```
CLS
FOR xpos = 0 TO GWIDTH STEP 100 LOOP
    COLOUR=RED
    VLINE( 0, GHEIGHT, xpos-1 )
    COLOUR=LIGHTPINK
    VLINE( 0, GHEIGHT, xpos )
    COLOUR=RED
    VLINE( 0, GHEIGHT, xpos+1 )
    UPDATE
REPEAT
END
```



WAIT

Purpose

Waits for the specified time to elapse.

Syntax

WAIT(*time*)

Description

This waits (does nothing) for *time* seconds. This may be a fractional number, but the accuracy will depend on the computer you are running it on, however delays down to 1/100th of a second should be achievable.

Example

```
REM COUNT 10 Seconds
CLS
seconds = 0
FOR i = 1 TO 10 LOOP
   WAIT( 1 )
   seconds = seconds + 1
   PRINTAT( 10, 10 ); seconds
REPEAT
PRINT "Elapsed "; TIME / 1000
END
```

WHEELDOWN

Purpose

Returns TRUE if the mouse wheel is scrolled down

Syntax

value = WHEELDOWN

Description

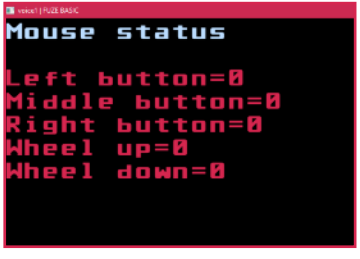
Returns a value of 0 or 1 (TRUE or FALSE) depending on whether it is press or not.

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
UPDATEMODE = 0
FONTSIZE (4)
LOOP
 CLS2
 INK = LIGHTBLUE
 PRINTAT (0, 0);  "Mouse status"
 INK = RASPBERRY
 PRINTAT (0, 2);  "Left button=";  LEFTCLICK
 PRINTAT (0, 3);  "Middle button=";  MIDDLECLICK
 PRINTAT (0, 4);  "Right button=";  RIGHTCLICK
 PRINTAT (0, 5);  "Wheel up=";  WHEELUP
 PRINTAT (0, 6);  "Wheel down=";  WHEELDOWN
 UPDATE
REPEAT
```



WHEELUP

Purpose

Returns TRUE if the mouse wheel is scrolled up

Syntax

value = WHEELUP

Description

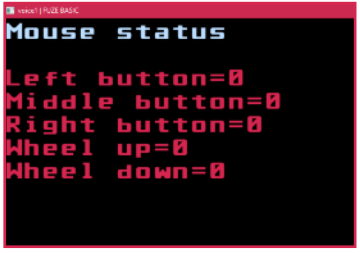
Returns a value of 0 or 1 (TRUE or FALSE) depending on whether it is press or not.

Associated

MOUSEOFF, MOUSEON, MOUSEX, MOUSEY, MOUSEBUTTON, GETMOUSE, SETMOUSE, LEFTCLICK, MIDDLECLICK , RIGHTCLICK, WHEELUP, WHEELDOWN, LOCKMOUSE

Example

```
UPDATEMODE = 0
FONTSIZE (4)
LOOP
 CLS2
 INK = LIGHTBLUE
 PRINTAT (0, 0);  "Mouse status"
 INK = RASPBERRY
 PRINTAT (0, 2);  "Left button=";  LEFTCLICK
 PRINTAT (0, 3);  "Middle button=";  MIDDLECLICK
 PRINTAT (0, 4);  "Right button=";  RIGHTCLICK
 PRINTAT (0, 5);  "Wheel up=";  WHEELUP
 PRINTAT (0, 6);  "Wheel down=";  WHEELDOWN
 UPDATE
REPEAT
```



WHILE REPEAT

Purpose

Loop while the specified condition is met.

Syntax

WHILE *condition* LOOP
  {*statements*}
REPEAT

Description

Execute the *statements* zero or more times while the *condition* is **TRUE** (Not **FALSE**). Because the test is done at the start of the loop the *statements* may not be executed at all.

Example

```
handle = OPEN( "whiletest.txt" )
FOR r = 0 TO 10 LOOP
   PRINT# handle,"Record "; r
REPEAT
CLOSE( handle )
handle = OPEN( "whiletest.txt" )
WHILE NOT EOF( handle ) LOOP
   INPUT# handle, record$
   PRINT record$
REPEAT
CLOSE ( handle )
END
```

Associated

BREAK, CONTINUE, LOOP, LOOP REPEAT, FOR REPEAT, REPEAT UNTIL, UNTIL REPEAT, WHILE REPEAT

Joystick and Gamepad commands

Purpose

Read input from specified Joystick or Gamepad

Syntax

See right

Description

LISTGAMEPADS returns a list of the recognised connected devices by name and **NUMGAMEPADS** returns the number of devices detected.

PRINT GETBUTTON (0, 0)

Will display the status of joystick 0 button 0.

Once you have detected a joystick is present then use **SETUPGAMEPAD(ID)** to use it.

listGamepads

Immediate mode only. Displays a list of connected devices.

numButtons(gamepad)

Returns the number of buttons on a specified device

numAxes(gamepad)

Returns the number of analog axis on a specified device

numHats(gamepad)

Returns the number of HATS on a specified device

numGamepads

Returns the number of devices connected

getAxis(gamepad, axis)

Returns the value of a given axis on a specified device

getButton(gamepad, button)

Returns the value of a given button on a specified device

getHat(gamepad, hat)

Returns the value of a given HAT on a specified device

SetupGamepad(gamepad)

Configure a specified device

Joystick and Gamepad

Example

```
UPDATEMODE = 0
FONTSIZE (3)
SETUPGAMEPAD (0)
LOOP
 CLS2
 INK = ORANGE
 PRINTAT (0, 0);  "The detected joystick has"
 INK = BRIGHTGREEN
 PRINTAT (0, 1);  "Buttons: ";
 INK = YELLOW
 PRINT NUMBUTTONS (0) - 1
 INK = BRIGHTGREEN
 PRINTAT (0, 2);  "Analog Sticks & Triggers: ";
 INK = YELLOW
 PRINT NUMAXES (0)
 INK = BRIGHTGREEN
 PRINTAT (0, 3);  "Digital pads: ";
 INK = YELLOW
 PRINT NUMHATS (0)
 INK = ORANGE
 PRINTAT (0, 5);  "Button states"
 FOR but = 0 TO NUMBUTTONS (0) - 1 LOOP
  INK = BRIGHTGREEN
  PRINTAT (but * 3, 6);  "#";  but
  INK = YELLOW
  PRINTAT (but * 3, 7);  GETBUTTON (0, but)
 REPEAT
 INK = ORANGE
 PRINTAT (0, 9);  "Analog states"
 FOR ax = 0 TO NUMAXES (0) - 1 LOOP
  INK = BRIGHTGREEN
  PRINTAT (ax * 8, 10);  "#";  ax
  INK = YELLOW
  PRINTAT (ax * 8, 11);  GETAXIS (0, ax)
 REPEAT
 INK = ORANGE
 PRINTAT (0, 13);  "Digital pad states"
 FOR hat = 0 TO NUMHATS (0) - 1 LOOP
  INK = BRIGHTGREEN
  PRINTAT (hat * 3, 14);  "#";  hat
  INK = YELLOW
  PRINTAT (hat * 3, 15);  GETHAT (0, hat)
 REPEAT
 UPDATE
REPEAT
PRINT
END
```

scanBackspace
scanTab
scanClear
scanReturn
scanPause
scanEscape
scanSpace
scanExclaim
scanQuoteDbl
scanHash
scanDollar
scanAmpersand
scanQuote
scanLeftParen
scanRightParen
scanAsterisk
scanPlus
scanComma
scanMinus
scanPeriod
scanSlash
scan0
scan1
scan2
scan3
scan4
scan5
scan6
scan7
scan8
scan9
scanColon
scanSemiColon
ScanLess
ScanEquals
scanGreater
scanQuestion
scanAt
scanLeftBracket
scanBackSlash
scanRightBracket
scanCaret
ScanUnderscore
scanBackQuote
scanA
scanB
scanC
scanD
scanE
scanF
scanG
scanH
scanI
scanJ
scanK
scanL
scanM
scanN
scanO
scanP
scanQ
scanR
scanS
scanT
scanU
scanV
scanV
scanW
scanX
scanY
ScanZ
scanDelete
scanKP0
scanKP1
scanKP2
scanKP3
scanKP4
scanKP5
scanKP6
scanKP7
scanKP8
scanKP9
scanKpPeriod
scanKpDivide
scanKpMultiply
scanKpMinus
scanKpPlus
ScanKpEnter
ScanKpEquals
scanUp
scanDown
scanRight
scanLeft
scanInsert
scanHome
scanEnd
scanPageup
scanPagedown
scanF1
scanF2
scanF3
scanF4
scanF5
scanF6
scanF7
ScanF8
scanF9
scanF10
scanF11
scanF12
scanF13
scanF14
ScanF15
scanNumLock
scanCapsLock
scanScrolLock
scanRShift
scanLShift
scanRCtrl
scanLCtrl
scanRAlt
scanLAlt
scanRMeta
scanLMeta
scanLSuper
scanRSuper
scanMode
scanCompose
scanHelp
scanPrint
scanSysReq
scanBreak
scanMenu
scanPower
scanEuro
scanUndo

Using the BBC micro:bit

Purpose

Allows control and interaction with the BBC micro:bit. Multiple devices can be connected at the same time.

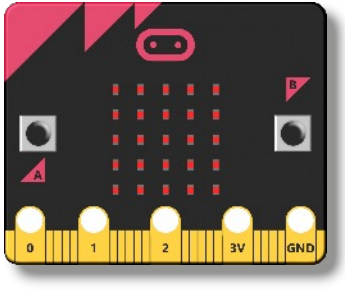
Syntax

See right

Description

Because FUZE Code Studio can interact with multiple devices including Ardunio's and BBC micro:bits it is advisable to initialise each device first. Please refer to DETECTDEVICES, DEVICETYPE and SETDEVICE.

If you only have one device attached then this is not required and the **SETDEVICE** statement is redundant. Otherwise **SETDEVICE** must be used before switching to a different device.



mbPRINT("text", speed)
Instructs the micro:bit to display the specified text at the given speed (0 very slow to 9 very fast). A text length of 20 or less is recommended when using Bluetooth communication.

mbPLOT(x, y, brightness)
Lights an individual LED at the specified position with a brightness between 0 (off) and 255.

mbCLS
Turns off all the micro:bit LEDs

mbLINE(x1, y1, x2, y2, brightness)
Draws a line of LEDs from x1, y1 to x2, y2 with a brightness between 0 (off) and 255

mbRECT(x1, y1, w, h, brightness, fill)
Draws a rectangle from x1, y1 with a width (w) and height (h) with a brightness between 0 (off) and 255. The 'fill' can be 0 (empty) or 1 (filled in)

mbBUTTONA
Returns the value of the micro:bit button A:
0 = not pressed 1 = pressed

mbBUTTONB
Returns the value of the micro:bit button B:

Using the BBC micro:bit

0 = not pressed 1 = pressed

MbTouch(pin)
Returns the capacitive value of the specified pin (0, 1 or 2)

value = mbCOMPASSHEADING
Returns the value of the micro:bit internal compass in degrees between 0 (north) and 359.

value = mbCOMPASSX / Y / Z
Returns the compass X, Y and Z values.

value = mbCOMPASSFIELDSTRENGTH
Returns the compass magnetic field value

value = mbACCELX / Y / Z
Returns the accelerometer X, Y and Z values (-2048 to +2048)

value = mbPitch
Returns the pitch value (-180 to 180)

value = mbRoll
Returns the roll value (-180 to 180)

value = mbTEMP
Returns the micro:bit's internal CPU temperature in degrees centigrade

value = mbGESTURE
Returns a value depending on the physical position of the BBC micro:bit. Values are as follows;

mbTiltUp = 1
mbTiltDown = 2
mbTiltLeft = 3
mbTiltRight= 4
mbFaceup = 5
mbFacedown = 6
mbFreefall = 7
mbshake = 11

mbDigitalWrite(pin, value)
Write a digital signal (0 or 1) to the specified pin.

mbAnalogWrite(pin, value)
Write a analog signal (0 to 255) to the specified pin.

mbDigitalRead(pin)
Reads the digital value (0 or 1) of the specified pin.

Using the BBC micro:bit

Example

```
rem display a sequence of effects
FOR counter = 0 TO 500 LOOP
MBPLOT(RND (5), RND (5), RND (255))
REPEAT
FOR counter = 1 TO 5 LOOP
FOR size = 1 TO 5 LOOP
MBRECT (0, 0, size, size, 255, 1)
WAIT (.05)
MBCLS
REPEAT
REPEAT
FOR y = 0 TO 4 LOOP
MBLINE (0, y, 4, y, 255)
WAIT (.1)
REPEAT
FOR y = 4 TO 0 STEP -1 LOOP
MBLINE (0, y, 4, y, 0)
WAIT (.03)
REPEAT
END
```

```
rem display and move a dot using the A & B buttons
mbx = 2
mby = 4
mbdir = 0
press = FALSE
MBCLS
MBPLOT (mbx, mby, 255)
LOOP
IF MBBUTTONA = 0 AND MBBUTTONB = 0 THEN press = 0
IF MBBUTTONA = 1 OR MBBUTTONB = 1 AND press = 0 THEN
press = TRUE
PLAYNOTE (0, 150 + (20 * mbx), 0.05)
MBPLOT (mbx, mby, 0)
IF MBBUTTONA THEN mbdir = -1
IF MBBUTTONB THEN mbdir = 1
mbx = mbx + mbdir
IF mbx < 0 THEN mbx = 0
IF mbx > 4 THEN mbx = 4
MBPLOT (mbx, mby, 255)
ENDIF
ENDIF
UPDATE
REPEAT
```

ARMBODY, ARMELBOW, ARMGRIPPER, ARMLIGHT, ARMRESET, ARMSHOULDER, ARMWRIST

Purpose

Move the robot arm body.

Syntax

ARMBODY(*direction*) or specified section
ARMLIGHT(switch)

Description

Activates the specified section of the robot arm according to the *direction* argument as follows:

1 Move clockwise (or light on)
0 Stop (or light off)
-1 Move anti-clockwise

Associated

ARMBODY, ARMELBOW, ARMGRIPPER, ARMLIGHT, ARMRESET, ARMSHOULDER, ARMWRIST

Example (ARM section)

```
REM Move the elbow for 1s anti-clockwise
ARMELBOW( -1 )
WAIT( 1 )
ARMELBOW( 0 )
END
```

Example (ARMRESET)

```
REM Move the elbow for 1s anti-clockwise
ARMELBOW( -1 )
ARMRESET
```

Example (ARMLIGHT)

```
REM Flash the LED with 1 second interval
LOOP
   ARMLIGHT( 1 )
   WAIT( 1 )
   ARMLIGHT( 0 )
   WAIT( 1 )
REPEAT
END
```

FUZE BASIC

fuze.co.uk