TOTALLY BASIC

A reference, cross-reference and substitution guide.

Covering over two dozen subsets on nearly a dozen platforms from **A** to **ZOrder**.

Compiled, Tested and Annotated by Earl R. Dingman

TOTALLY BASIC

A Reference, Cross-Reference and Substitution Guide.

Covering over two dozen subsets on nearly a dozen platforms from **A** to **ZOrder**.

Direct support for:

AMIGA: Amiga Basic, Hi Soft BASIC and True BASIC. **Apple II**: Integer BASIC and Apple Soft BASIC. **Atari 8 bit**: Atari XL BASIC. **Atari ST**. ST BASIC 1.0/2.0, Hi Soft BASIC, BBC BASIC, FAST BASIC and GfA BASIC. **Commodore 8 bit**: PET, C-64 BASIC and C-128 BASIC. **CP/M**: MBASIC and ABASIC. **Macintosh**: MS BASIC, Z BASIC, Future BASIC, VIP BASIC and True BASIC**. Mainframes**: ANSI Minimal BASIC. **PC DOS**: Business BASIC II. BASICA, IBM BASIC, MS BASIC, GW BASIC, QBASIC, Quick BASIC, Turbo BASIC, Power BASIC, True Basic, Microsoft Professional Development Basic 7.0 and Visual BASIC. **PC Windows**: CA REALIZER, Visual BASIC 1 through 6 Professional. **Psion:** OPL. **Tandy/Radio Shack**: TRS-80 BASIC, Color Computer BASIC and Tandy BASIC for PC. **Timex - Sinclair**: 100 BASIC. Spectrum Color BASIC. **VAX**: DCL BASIC.

ACOS()

(GfA BASIC 3.0 for Atari ST, True BASIC for Amiga, Macintosh, PC and Unix)

Provides the **A**rc **COS**ine for the value placed inside the parentheses.

Syntax: **PRINT ACOS(n)** or X = **ACOS(n)**

Substitutions: Generally use a **DEF FN** or **DEFFN** function:

DEF FNACOS(n) = ATN(n / SQR(-n * n+1)) + 1.5708

Then you would use the created function in the program: **FNACOS(n)**.

Also see: **DEF FN** and **DEFFN** plus **OPTION ANGLE ACS**

SUBSTITUTION FACTOR: Excellent **PORTABILITY:** None

ACS

(Timex/Sinclair BASIC)

Returns the **A**rc **C**o**S**ine of a value in radians.

Syntax: **PRINT ACS** x or A = **ACS** x

Substitutions:: **DEF FNACS(X) = ATN(X / SQR(-X * X + 1)) + 1.5708** And then use: A = **FNACS(X)** or **PRINT FNACS(X)** inside the program to replace **ACS.** Also see: **DEF FN** and **DEFFN** and **ACOS()**

SUBSTITUTION FACTOR: Fair to Good **PORTABILITY:** None

Action

(Visual BASIC)

A property used with dialog and message boxes on Visual BASIC 1.0.

syntax: **CMDialog.Action = n**

Where **n** is either 0 (nothing), 1 (open files), 2 (save files), 3 (color), 4 (fonts) and 5 (printer). ACTION is one of many various parameters including **Filter,Filename, etc.**

Usage: **CMDialog.Filter= "Images |*.jpg;*.gif"**

Additionally, Visual BASIC 3.0 Professional Edition uses **Action** to work with OLE (Object Linking and Embedding) 2.0 in the following syntax:

OLEControlName.Action = x

Where **ControlName** is one of the following: 0 (creates an embedded object), 1 (creates a linked object), 4 (sends data to the clipboard), 5 (copies data back from clipboard), 6 (gets data from linked file), 7 (activates the object), 9 (closes an embedded object), 10 (deletes an object), 11 (saves linked object data), 12 (loads linked object file), 14 (displays dialog box for inserting), 15 (provides a pasting dialog box), 17 (returns supported linked application **"Verbs"**) and 18 (used for OLE 1.0 compatibility).

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None to Poor

ADD

(GfA BASIC for Atari ST)

Adds the second variable or constant, to the first variable.

Syntax: **ADD** a, b

Where "b" is added to "a".

Substitutions: A = A + B.

SUBSTITUTION FACTOR: Excellent **PORTABILITY**: None

AddItem

(Visual BASIC for Windows)

Adds an item to a *combo box* list and additionally under Visual BASIC 3.0 adds an item to a spreadsheet grid. The syntax is:

Grid_Or_ListName.**AddItem L$,I%**
or
ListName.**AddItem =** "A name or string, etc."

Where *L$* is a string variable you previously declared and assigned and I% is an optional position in the list (default is the end of your list or grid).

Substitutions: Use a common array, see: DIM However list boxes and sheets must be created manually of with third party tools in most other BASICs.

SUBSTITUTION FACTOR: Good **PORTABILITY:** None

AddNew

(Visual BASIC 3.0)

Data base function used in conjunction with **Update** that is compatible with Microsoft Access. Moves the current record to the end and passes new data to the Access engine copy buffer

Syntax: Name.**RecordSet.AddNew**

Where *Name* is a user defined label. Also see: **Update.**

Substitutions: Use an array (see: **DIM**) or a sequential file set for append (see **OPEN**) or a random access file using order placement calls (see: **OPEN**). Note, most other BASICs do not support ACCESS or any other data base so you must create your own data base engine.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None

ADDRIN
ADDROUT

(GfA BASIC For Atari ST)

A four byte starting address for Atari ST GEM AES functions. Values are put in using a **LPOKE** command.

Substitutions: Rough equivalents include PC "interrupts" or Windows API calls and Macintosh "Tool box" routines. AES routines include alert boxes, drawing primitives, drive information, etc. Also see: **GEMSYS, GCONTROL, GEMCONTROL** and **VDISYS**

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None

AddSys()

(CA Realizer)

Adds additional search paths (sub directories or file folders) to the default paths for finding files.

Syntax: **AddSys(T,"path")**

Where **T** is one of two system constants: **_LoadDir** or **_MacroDir** and "path" is the standard format for drive and directory assignment (e.g. "C: \MYFILES").

Substitutions: May not require a subsitution on other systems or you can assign paths to string variables and create routines or algorithms to check the various paths. Some Microsoft BASICs such as Q, Quick and Visual BASIC can also use **CurDir** to assign a path.

SUBSTITUTION FACTOR: **PORTABILITY**: None

AddToDate()
AddToTime()

(CA Realizer)

Adds (increases) a value to a date or time setting.

AddToDate(CD,Y,M,D,H,M,S)

Adds any combination of date, month, year, hour, minutes or sections to the current values held in memory where CD is the current date and the rest of the values are optional parameters that will be added to the CD figure as either Years, Months, Days, Hours Minutes or Seconds.

AddToTime(CT,H,M,S)

Adds only the time values of Hours, Minutes or Seconds to the Current Time value CT.

Substitutions: VB has various date options, see: **DateSerial(year, month, day)**

Older computers store date and time as a string and you must use **VAL** to convert and then add a new value in seconds or fractions of a second, see **DATE, DATE$, TIME**, **TIME$, DI$, TI$, VAL**. Some computers have no clock settings.

SUBSTITUTION FACTOR: None to Fair **PORTABILITY**: None

AFTER
AFTER CONT
AFTER STOP

(GfA BASIC 3 for Atari ST)

A type of event driven control that is based upon units of time (ticks).

Syntax: **AFTER x GOSUB lable_name**
REPEAT
UNTIL event_parameter

After "x" amount of ticks you direct the program to a **GOSUB** until some "event_parameter" has occurred. **AFTER STOP** allows you to suspend the action of **AFTER** and **AFTER CONT** allows it to continue once more.

Substitutions: ON...events (**ON TIMER GOTO) TIMER**.**GOTO** or **GOSUB**.

SUBSTITUTION FACTOR: None to Fair **PORTABILITY**: None to Poor

ALERT

(GfA BASIC For Atari ST and MS PD BASIC 7.1)

Calls a system "Alert box" (graphical box designed for interaction with the user)

MS PD BASIC 7.1 Syntax: **X%=ALERT(s%,t$,tr,lc,br,rc,ba$,bb$,bc$)**

Where X% returns the button selected value; s% is either 1 (fixed line length), 2 (word wrap), 3 (word occurring at spaces between the word nearest the right) and 4 (centered text); t$ is the text to be placed on each line with a vertical bar | used to separate lines; tr is the top row position of the upper left corner for the alert box; lc is the left column position of the box, br is the bottom row position for the lower right corner of the alert box rc is the right column of the alert box, ba$, bb$, bc$ are the text strings that go inside each of the three buttons.

GfA BASIC for Atari ST syntax is: **ALERT** I%,T$,DB%,BT$,X%

Where **I%** selects the ICON (0 = none, 1 = !, 2 = ?, 3 = stop), **T$** is your text to tell the user what the problem is or what to do (up to 4 lines of 30 character text, with a vertical bar [|] used to terminate each line), **DB%** is the default button that works from the <**RETURN**> key (0 = no default button, 1, 2 or 3 assigns one of the three possible buttons ad default), **BT$** is the text found in each button (e.g. OK) with the vertical bar used to separate up to three buttons, text length is very limited, and **X%** is the value returned from the button press (1, 2 or 3) which is passed to your program to send you to a new routine.

Substitutions: FNform_alert(DB,T$), **MsgBox$(m$,b%,t$)** , **GB, GCONTROL, GEM_CONTROL(**) and **GEMSYS.**

For Visual BASIC add a new form to the project and put whatever buttons, text and actions that are required.

All others must created manually with code.

SUBSTITUTION FACTOR: Poor to Good **PORTABILITY:** None

Align
Alignment

(Visual BASICs for Windows)

Align is used to position a ***Picture Box***. Generally set from the menu to a value of 0 (default setting which allows placement of the box anywhere you choose), 1 (used with Toolbar, puts the box flush to the top) and 2 (use with Status bar and places the box flush with bottom).

Alignment is used to position text labels. 0 is left justified, 1 is right justified and 2 is centered.

Substitutions: (see **PUT, GSHAPE, BOX SHOW** and **BLOAD**), **PICTURE FIELD**. **CBOX, RBOX** and **LBOX**. **vst_alignmen**t which has a value of 0 for left justified, 1 for centered and 2 for right justified. Text is then placed using **v_gtext. TEXT, ACHAR), CHAR, LOCATE, GOTOXY, PRINT AT, PRINT@, @** you must devise algorithms to center and align.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None to Poor

ALINE

(GfA BASIC version 3 for Atari ST)

An alternative method of drawing lines using the Atari drawing primitive.

Syntax: **ALINE x, y, xx, yy, c, s, v**

Where **X, Y, XX** and **YY** are the starting and ending coordinates, **C** is a color value from 0 - 15, **S** is type of line (solid, dot, dash, dot, dash) and **V** is displacement value (0 replace, 1 transparent, 2 inverted, 3 inverted transparent).

See: **v_pline LINEF LINE, PSET, SET, PLOT** or **DRAW PSET** and **SET**

SUBSTITUTION FACTOR: Fair to Good **PORTABILITY:** None

ALL

(Interpreter BASICs for IBM PC and Atari ST BASIC)

Used with **CHAIN** and possibly with **MERGE** to retain and pass the values of all variables between the old and the new code segments. Helpful in breaking the 64 Kb barrier found in PC programs. Also runs “Overlays” and switches between program code files.

Syntax: **CHAIN** "filename.BAS",x,**ALL**

CHAIN MERGE can also be used, where some lines can be retained from the original program (**CHAIN** without merge simply replaces all lines from the original program with the new program).

Substitutions: SUB SEGMENT BANK, **RUN, SWAP** and **FETCH**. BASICs with no memory, file, array or segment limitations has no pressing reason to use this call.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** Poor

AnalyzeChart()
AnalyzeChartMS()
AnalyzePie()
AnalyzeScatter()
AnalyzeScatterMS()

(MS PD BASIC 7.1)

Part of an internal library of graphics charts for making presentations (pie, bar, column, line and scatter). Requires that the library (**CHRTBEFR.QLB**) be loaded. The following should be at the top of your code: **REM $INCLUDE: 'CHART.BI'**

Allows you to change the values or parameters in an existing **Chart()**. The parameters, values and labels are identical as those you created. See **Chart()** for more details.

Substitutions: Chart() Chart.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None

AND

Found in virtually all systems. Missing from Apple Integer BASIC, most ANSI Minimal BASIC subsets and TRS-80 Level I BASIC. May also be missing or supported differently with some GfA subsets for Atari ST. A "logical" operator used to make compound comparisons where both values must be compared.

General syntax is: **IF** x = y **AND** u = w **THEN**....

If only one side of the equation is true, the "**THEN**" statement will **NOT** be activated. Both x = y **AND** u = w must be true for the "**THEN**" to occur. (GfA BASIC doesn't support THEN, but the concept is the same.)

Some modern BASIC subsets additionally use **AND** to designate how the graphics will appear on the screen in certain graphics commands, such as **PUT** or **SSHAPE**.

SUBSTITUTION FACTOR: Excellent **PORTABILITY:** Very Good

AND()

(GfA BASIC version 3 for Atari ST)

Compares two numbers to see which bits are set the same and return a corresponding value.

Syntax: **X = AND(y,z)**

In which **Y** and **Z** are compared and the similar bits are given to **X**.

Substitutions: Many of the very modern BASICs (Q BASIC, Visual BASIC, GfA BASIC, etc.) support the following syntax: **X = Y AND Z** for obtaining the same results. Older BASICs (generally pre 1988) have no substitutions except to create a very extensive algorithm that compares the bit places of both numbers.

SUBSTITUTION FACTOR: Poor to Fair **PORTABILITY**: None

ANGLE()

(True BASIC for Amiga, Macintosh, PC and Unix)

Returns the angle (in either degrees or radians, depending on your setting of **OPTION ANGLE**) between two specified points in the parentheses. The range is between -180 and +180 degrees (or radian equivalent) and is in a counter (or anti) clockwise direction.

Syntax: **X = ANGLE(A,B)**

Substitutions: Other BASICs must create a manual algorithm to duplicate this function.

SUBSTITUTION FACTOR: Good **PORTABILITY:** Poor

AnimateCells()
AnimateControl()
AnimateFrame()
AnimateSelect()
AnimateSpecialFrame()

(CA Realizer)

Used to animate bit map

AnimateCells("filename")

Where you animate either the named bitmaps/CA Realizer Pictures or the numbered cells placed in sequence. Where "filename" is any legal bit map or picture extension saved to disk.

AnimateCells("aniseq#.bmp",1,25)

Where, in this example, 25 different pictures will be displayed from 1 to 25 and the pound sign (#) is used to put the numbers into this sequence.

AnimateControl(_systemConstant)

The "**_systemConstants**" are used to: **_Start _Stop _Clear _Reset _SetSpeed _SetOffset**
AnimateFrame(CN,X,Y,M)

Where CN is the cell number (in the previous example a separate frame must be made for each of the 25 bit mapped pictures), **X** and **Y** are the pixel positions and **M** is a value of time in milliseconds.

AnimateSelect(An,Fn)

This put "focus" on a specified animation number An in the specified Form number.

AnimateSpecialFrame(_systemConstant)

An **EOF** flag used for animation work that allows you (with system constants) to **_Stop** when the animation reaches this point **_Notify** the Form when the animation reaches this point or to **_Pause**;M where **M** is time in milliseconds -- only **_Pause** adds an extra parameter.

Substitutions: PUT() and **GET()**, **SSHAPE** and **GSHAPE**

SUBSTITUTION FACTOR: None to Good **PORTABILITY**: None

ANY

(MS PD BASIC 7.1)

Disables "type" checking when passing records to a procedure.

Syntax: **DECLARE SUB Name_of_Record (variable AS ANY)**

You must also use **TYPE** to define the record and **DIM** the name of that type, which is then put into the parentheses of **Name_of_Record(name_of_type)**.

Substitutions: There are generally no substitutions for this process

SUBSTITUTION FACTOR: None **PORTABILITY:** None

APOLY TO

(GfA BASIC version 3 for Atari ST)

Creates a filled polygon using an Atari ST library primitive. It is generally easier to use standard drawing functions like **LINEF** and **FILL**.

Syntax: **APOLY *xy(0), np, h to l, c, v, *p, pn**

Where ***xy(0)** is a pointer to the address of an array of X and Y coordinates, **NP** is the number of points to be plotted (should be the same as half the number of active elements in array **XY**), **H** is the highest point on the screen to start a fill and **L** is the lowest point on the screen to start a fill, **C** is the color, **V** is the verb (0, 1, 2 or 3) that determines how the pixels are displaced**, *P** is the pointer to the address of variable that contains the pattern information for the lines styles and **PN** is a number pattern, generally an odd value. See **ALINE** for more details on some of these parameters.

Substitutions: v_fillarea n p().**AREA**. **AREA** and **AREA FILL, LINE**, **PLOT**, **DRAWTO, FILL** or **PAINT**

SUBSTITUTION FACTOR: None to Good **PORTABILITY**: None to Poor

App
AppActivate

(Visual BASIC 3.0)

Allows you to activate any standard windows application that is already running (you can run any application using **SHELL**) and brings it "into focus."

Syntax: **AppActivate** NameOfTheWindowApplicaton$

Where the Name...$ is the same as on the title bar (it is not case sensitive, but spelling and spaces must be exact). You can also pass keystrokes to the application using **SendKeys** and get a return indicator from the application you have called.

App an object used to return the path or name of the .EXE file.

Syntax: **App.Path**
App.EXEName

Substitutions: None really, although see: RUN, SHELL, CHAIN and APPEND for possibilities

SUBSTITUTION FACTOR: None to Very Poor **PORTABILITY:** None

APPEND

A). (VAX DCL BASIC)

Melds code lines saved in a file into the code lines of the program you're currently working in memory.

Substitutions: CHAIN, RUN, App, AppAcitivate, MERGE

SUBSTITUTION FACTOR: Good **PORTABILITY:** None

B), (Commodore C-128 BASIC 7.0 & Some Apple DOS versions)

This is a file function that allows you to add more data to an existing disk file.

The Apple syntax is: **APPEND** filespecs, **S**x, **D**y, **V**z

Where filespecs is generally a name.

Commodore C-128 syntax is: **APPEND #**n,"name",**D**x

Where "**n**" is the channel number, "name" is the filename written as text inside quotes or as a string variable, the letter **D** must be printed out, followed by a drive number (which replaced "x").

Substitutions: OPEN "A", #n,"filename" or **OPEN "**Fi**lename" FOR APPEND AS #**
Commodore C-64 and PET use: **OPEN filenumber, dn, secondary, "filename".** The **dn** is 1 for cassette or 4 for disk drive. Also see **DOPEN** and **OPEN** for more details. Mini and

Mainframe computers tend to use **FILE** or **FILES**. GfA BASIC for Atari ST also supports **BPUT**. Future BASIC also supports **APPEND** under a separate listing.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None to Poor

C).(FutureBASIC for Macintosh)

Sets the file pointer to the last entry in a file so that more entries can be added to the file

Syntax: **OPEN "R", #1,"TEST.FIL"**
APPEND #1

Substitutions: See **APPEND** for the Commodore C-128 and Apple DOS **OPEN** "TEST.FIL" **FOR APPEND AS #**1) **FIELD, PUT** and **PRINT LOC, EOF, LOF FILE** or **FILES**

SUBSTITUTION FACTOR: None to Very Good **PORTABILITY:** None

APPL_EXIT()
APPL_FIND()
APPL_INIT()
APPL_READ()
APPL_TPLAY()
APPL_TRECORD()
APPL_WRITE()

(GfA BASIC version 3 for Atari ST)

AES library calls that are not generally used **APPL_EXIT()** A dummy function, use **SYSTEM** instead. **I& = APPL_FIND("name")** Searches for the application already in memory (generally only desk accessories and GfA BASIC unless you are using a newer Atari ST with multi-tasking operations) and returns its application identification number or 65535 (nothing found).**I& = APPL_INIT()** Returns the application ID of your program.**X& = APPL_READ(I&, NB, *B)** Reads **NB** bytes from an event buffer **B** (an address pointer is specified) for the application identified as **I&**. **APPL_WRITE(I&,NB, *B)** is used to put bytes into a buffer using the same parameters.**APPL_TRECORD(M,N)** records events like mouse activity and **APPL_TPLAY(M,N,S)** plays them back at a designated speed **S** (1 to 10,000). These functions do not work very well. They return data in 8 bytes.

Substitutions: Macintosh ToolBox routines, Windows API calls, DOS Interrupts.

SUBSTITUTION FACTOR: None to Fair **PORTABILITY**: None

APPLE MENU

(FutureBASIC for Macintosh)

Allows you to put one or more lines of text under the "Apple" menu heading (above where the Desk Accessories are located on the far left menu item, which is the Apple logo

Syntax and Usage: **APPLE MENU** "About This Program"
APPLE MENU "About This Program;Registration"

In the last example, two separate entries are generated (the semicolon is what the compiler looks for to separate the various entries). Most programs rarely use a second or third entry.

This function is primarily for advanced programmers writing commercial programs for the Macintosh.

Substitutions: Only Macintosh, Amiga and Atari ST use the menu bar concept for "desk" accessories (which are direct equals to IBM PC *TSR* (*Terminate and Stay Resident*) programs, except that the PC uses hot keys or macros to activate TSR programs and not a menu item. **CHAIN...MERGE...ALL CALL INTERRUPT**, **fnMENU&(menu_string**$) and **menu_bar tree&,flag MENU** m,v and **MENU(x**) to

SUBSTITUTION FACTOR: None to Very Good **PORTABILITY:** None

AREA

(Atari ST BASIC 2.0)

Creates a filled polygon.

Substitutions: Amiga BASIC, Hi Soft BASIC for Amiga uses **AREA** and **AREAFILL**. GfA BASIC for Atari ST uses **POLYFILL pts, xp(),yp()** or (version 3.0) **APOLY TO.** Hi Soft Professional BASIC uses the VDI library call **v_fillarea n, pts()**.

A "polygon" routine generally uses a mouse to pick out points (pixels) on the screen, these are then connected together with lines and the interior is filled. Most BASICs not supporting a direct "polyfill" type routine must use standard **LINE** (PC), **LINEF** (ST BASIC 1.0), **PSET** (Macintosh, PC, Visual BASIC, ST BASIC 1.0, Hi Soft), **PRESET** (PC, ST BASIC 1.0, Hi Soft), **SET** (TRS-80), **PLOT** (Future BASIC), **PLOT TO** (Future BASIC, Apple), **DRAW, DRAW TO** (Atari XL) routines with coordinate obtain with a mouse or manually written by the programmer, then use a standard fill routine (**PAINT** for the PC, **FILL** for the Atari, multiple **LINE** statements must be used with Visual BASIC which doesn't support either **FILL or PAINT**) within the boxed area.

Atari ST, Macintosh and possibly Visual BASIC and CA Realizer may be able to use graphics primitives found in various system files (MS Windows API,GDI), libraries (Atari VDI) or (Macintosh) **TOOLBOX**

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None

AREA and AREAFILL

(Amiga BASIC, Hi Soft BASIC for Amiga)

Draws a polygon (3+ sided figure) at specified coordinates.

Syntax is: **AREA (x,y)**
Optional usage: **AREA STEP (x,y)**
and
AREAFILL z

Where **STEP** is a number that moves the drawing tool a designated "step." For **AREAFILL** the "z" is either 1 (colors are inverted) or 0 (the designated PATTERN is used.

Also see: **PATTERN** and **STEP**

Substitutions: GfA BASIC for Atari ST **POLYLINE, POLYFILL** and **POLYMARK**, also **APOLY TO.** Atari ST BASIC 2.0 **AREA.** Hi Soft Professional BASIC the library call **v_fillarea**. Most other

BASICs **PLOT, PSET, PRESET, LINE, LINEF, FILL, PAINT.** There is no substitute for older computers with no or limited graphics, such as CP/M machines using M or S BASICs.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None

ARECT

(GfA BASIC version 3 for Atari ST)

Same as **PBOX**.

Syntax: **ARECT x,y,xx,yy,c,v,*p,np**

Where **X, Y, XX** and **YY** are the starting and ending coordinates, **C** is the color from 0 - 15, **V** is the verb from 0 - 3 (0 is replace, 1 is transparent, the other are inverse), ***P** is an address pointer to a line pattern value(dot, dash, etc.) and **NP** is the pattern number (generally an odd number).

Substitutions: Macintosh **TOOLBOX** routines. Hi Soft Professional BASIC library call **vr_recfl x, y, xx, yy**. Any computer capable of graphics calls **LINE, PLOT, PLOT TO, DRAW TO**, **PSET, PRESET PLOT, PLOT TO, DRAW TO**, **PSET, PRESET, FILL** or **PAINT**. Computers using ASCII based drawing systems usually have special effects ASCII characters such as stars, half boxes, diamonds, etc. This can be used to created a rectangular area of a pattern.

SUBSTITUTION FACTOR: None to Good **PORTABILITY**: None

Arrange

(Visual BASIC 3.0)

Used in conjunction with CONSTANT.TXT file. Allows you to position (arrange) child windows inside a master parent window (technically called an MDI or Multiple Document Interface).

Syntax/usage is: **MDIForm1.Arrange CASCADE**

You can also **ARRANGE_ICONS** and **TITLE_HORIZONTAL** using this syntax and replacing **CASCADE**. **Form1** is the default setting which you can rename when you create your parent MDI form (window).

Substitutions No real need for substitutions.

SUBSTITUTION FACTOR: None required **PORTABILITY:** None

ARRAY SORT
ARRAY SCAN
ARRAY INSERT
ARRAY DELETE

(Power BASIC 2.1)

Performs complex functions to arrays in whole or part.

Numeric syntax:

ARRAY SORT a(fe) FOR c, TAGARRAY b(), DECEND

String syntax:

ARRAY SCAN a$(fe) FOR n, FROM s TO e, COLLATE UCASE, expression, TO ep

Sorts or scans elements from same size array **a()** into array **b()** Parameter **fe** beginning point in the default is the first element, **n** is the number of elements to sort (whole array by default). **ASCEND** or **DECEND** is the direction to sort. **COLLATE** (**UCASE** or **string**) weights either as uppercase only or both upper and lower **FROM s TO e** is an option to sort the entire string or just a few characters into the string.

ARRAY INSERT a(e) FOR n, x
ARRAY DELETE a(e) FOR n, x

The parameters are the same for both of these operations and works for either string or numeric arrays. The parameter **e** is the number of the element that is to be "inserted" or "deleted" (where in the array to put or take a value), while n is the number of elements to shift (if omitted all elements from **e** to the end of the array will be shift up or down, depending on whether you are making an I**NSERT** or **DELETE** operation) and **x** the value to be inserted into the element (for I**NSERT** this value will be placed at position **e**, for **DELETE** this value will be placed at the end of the array or the end of the number of elements to shift, if omitted in a **DELETE** it will be zero or nil by default).

Substitutions: Future BASIC uses **INDEX F**. CA Realizer uses for string **ARRAY SCAN**.

Other BASICs use a standard sort routine for **ARRAY SORT**. For **ARRAY SCAN** an **INSTR** (**CPOS, POS()**)keyword (or substitution) can be used with a **FOR TO** or **WHILE WEND** loop and an **IF** logical comparison or test each element of the array. For **ARRAY INSERT** and **DELETE** you must **DIM** new arrays (with more elements for **ARRAY INSERT** or the same number for **ARRAY DELETE**) and use a loop with a logical **IF** to find the position in the array to make the insertion.

SUBSTITUTION FACTOR: Excellent **PORTABILITY:** None to Very Poor

ARRAYFILL

(GfA BASIC For Atari ST)

Fills all elements of a numeric array with a constant value in all elements

Syntax: **ARRAYFILL** array_name(),constant

Substitutions: For ANSI Minimal BASIC **see ZER**, **CON, IDN and MAT.** For most other BASICs (and to put a value other than zero or one into an ANSI Minimal BASIC array) a **LOOP** must be used.

SUBSTITUTION FACTOR: Excellent **PORTABILITY:** None

ARRPTR

(GfA BASIC For Atari ST)

Returns the address of either a string or an array.

Syntax: A%=**ARRPTR(B$)**
B%=**ARRPTR(**C)**)**

Substitutions: VARPTR, **VARPTR$** or **SADD**.

Future BASIC try **LINE** or possibly **@array_name()**.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None

AS

(Q BASIC, Quick BASIC and Visual BASIC)

Used with **TYPE, COMMON, SHARED, DECLARE, SUB, FUNCTION, FORM, CONTROL** and **DIM** to declare variables **AS** a particular value type (integer, string, etc.).

Syntax is: **DIM** X **AS** Integer

Substitutions: Most older BASICs only support numbers (plain variable) or strings (with a dollar sign after the variable): X and Z$. Also see: **DEFINT**, **DEFDBL**, **DEFSGL**, **DEFSTR**.

UBSTITUTION FACTOR: None to Fair **PORTABILITY:** None

ASC()

Found in most subsets. Missing from Apple Integer, Timex/Sinclair and TRS-80 Level I BASIC and True BASIC. Returns the ASCII code value for the first letter in a string.

Syntax is: **PRINT ASC("A")**

Substitutions: Visual BASIC **_KeyPress** Timex/Sinclair/Spectrum supports **CODE** Most Atari ST BASICs **A = INP(2)** True BASIC uses **ORD()** Power BASIC supports ASCII().

SUBSTITUTION FACTOR: None Required to Good **PORTABILITY:** Very Good

ASCII()

(Turbo/Power BASIC)

Almost identical to ASC(), except ASCII() supports a null string (returning -1) while ASC() does not (and returns no value)

Syntax: **ASCII("string expression")**

Substitutions: ASC() for most other BASICs, **CODE** for Timex/Sinclair Spectrum**, KeyPress** for VB

SUBSTITUTION FACTOR: Good **PORTABILITY:** Very Poor

ASIN()

(GfA BASIC 3.0 for Atari ST and True BASIC for Amiga, Macintosh, PC, Unix)

Provides the **A**rc **SIN**e (X) of a value (V)

Syntax: **X** = **ASIN(V)** or **PRINT ASC(V)**

Substitutions: Timex/Sinclair uses: **ASN v**. Other BASICs: **DEF FNASIN(v) = ATN(v / SQR(-v * v + 1))** then use **FNASIN(v).** Also see **DEF FN or DEFFN.**

SUBSTITUTION FACTOR: Excellent **PORTABILITY:** Very Poor

ASK ... or ***ASK# ...***

(True BASIC for Amiga, Macintosh, PC and Unix)

Some specific functions are not supported by some of the computer systems listed above. SUBSTITUTIONS for other BASICs accompany them.

ASK BACK
ASK COLOR
ASK COLOR MIX
ASK CURSOR
ASK FREEMEMORY
ASK MARGIN
ASK MAX COLOR
ASK MAX CURSOR
ASK MODE
ASK PIXELS
ASK SCREEN
ASK TEXT JUSTIFY
ASK WINDOW
ASK ZONEWIDTH

The following are used with the True BASIC disk file system:

ASK#x:ACCESS
ASK#x:DATUM
ASK#x:DIRECTORY
ASK#x:ERASABLE
ASK#x:FILESIZE

ASK#x:FILETYPE
ASK#x:MARGIN
ASK#x:NAME
ASK#x:ORGANIZATION
ASK#x:POINTER
ASK#x:RECORD
ASK#x:RECSIZE
ASK#x:RECTYPE
ASK#x:SETTER
ASK#x:ZONEWIDTH

SUBSTITUTION FACTOR: None To Excellent **PORTABILITY:** Very Poor

ASK MOUSE

(Atari ST BASIC 2.0)

Obtains the coordinates and button states of the mouse. Buttons states are 0 for no buttons 1 for left, 2 for right and 3 for both.

Substitutions: Only very modern computers support a "mouse." A "joy stick" command can be substituted on some computers to replace the mouse. Other computers must replace mouse routines using a command line (menu choices picked by entering a number or letter) or by moving the cursor using key presses and then making a choice by using the <RETURN> or <ENTER> key. Hi Soft BASIC for Atari ST, Amiga BASIC, Hi Soft BASIC for Amiga and Macintosh BASIC use: **MOUSE** or **MOUSE()**. GfA BASIC for Atari ST uses **MOUSEX, MOUSEY, MOUSEK. F**uture BASIC for Macintosh uses **MOUSE(_horz), MOUSE(_vert), MOUSE(_down)** to get the x,y and button states. Visual BASICs uses: **Object|Form_MouseUp(), _MouseDown(), _MouseMove()** event

PC DOS Quick and Power BASIC users must call Interrupt 31. GW BASIC uses a machine language routine to access the mouse.

SUBSTITUTION FACTOR: None to Good **PORTABILITY:** None

ASK RGB

(Atari ST BASIC 2.0)

Returns the color mix for each of the various color indexes or registers.

Syntax: **ASK RGB** cr,r,g,b

Where cr is the color register, r,g,b are the values of red, blue & green.

Substitutions: Hi Soft Professional BASIC can use the VDI library call **vq_color i,f,rgb().** GfA BASIC and Atari ST BASIC 1.0 can use the VDISYS routine op code 104. See *Atari ST Section* for more details.

True BASIC can use **ASK COLOR MIX.**

Visual BASIC and Future BASIC both support up to 16.7 million colors per pixel (limited to your color board and monitor), therefore you generally need to get a color mix for a specific, individual pixel. Also you can declare colors as being either background or foreground in nature (the distinction is seen when using drawing tools that have displacement verbs like **AND, OR, XOR, PSET,** etc.). Visual BASIC uses **RGB(r%,g%,b%)** to get the individual values.

Future BASIC can use the **ToolBox** routine **FN GETCOLOR** (Quick BASIC calls this as the **TOOLBOX** routine **GETCOLOR** also **CALL RGBFORECOLOR(**rgbCOLOR**)** and **CALL RGBBACKCOLOR(rgbCOLOR)**. All other systems have fixed color (with a limit of from 2 to 16 total colors possible) parameters that can't be mixed or altered.

POINT() can often be used to return the color value of an individual pixel.

Timex/Sinclair Spectrum Color BASIC uses **ATTR()**. Apple Iic uses **SCRN**.

SUBSTITUTION FACTOR: None to Fair **PORTABILITY:** None

ASN

(Timex/Sinclair BASIC)

Returns the **A**rc **S**i**N**e of a value in radians. Syntax: A=**ASN** x Where A is the Arc SiNe of x.

Substitutions: The **DEF FN** function can be used: **DEF FNASN(X) = ATN**(X / **SQR**(-X * X + 1)) Then use: A = **FNASN(X)** as a substitution for A=**ASN** x in your programs.

True BASIC and GfA BASIC use **ASIN(X)** as a substitute.

SUBSTITUTION FACTOR: Good **PORTABILITY:** None

AT

(TRS-80 Level I BASIC and GfA BASIC for Atari ST)

See **@** for more details.

For the TRS-80 this function, used in conjunction with **PRINT**, places the printed text or cursor at a specific screen location designated by a number from 0 to 1023 (23 is used in the below example).

Syntax is: **PRINT AT**23 "Hello"

For GfA BASIC this functions requires a row and column value and the syntax is: **PRINT AT(r,c)**

Substitutions: Very old subsets can only use **TAB()** or **TAB** to place the cursor at a designated column. **SPC()** can be used by some subsets. Apple uses **VTAB** and **HTAB** for r and c or x and y coordinates, respectively. Most modern subsets use like Amiga, Commodore C-128, Macintosh, Hi Soft BASIC for Atari ST all subsets for the IBM PC use **LOCATE** x,y. Atari ST BASICs use **GOTOXY** y,x. Atari XL uses **POSITION** r,c Commodore C-128 also uses **CHAR.** GfA BASIC can also use **TEXT.** The VAX DCL uses: **ESC + "["+STR$(X)+";"+STR$(Y)+"H".** Some older BASICs require that you **POKE** values into certain locations to obtain a substitute.

SUBSTITUTION FACTOR: None to Very Good **PORTABILITY:** Poor

ATEXT

(GfA BASIC version 3 for Atari ST)

Similar to **ACHAR** except that only position and font can be specified, no color, effects or angle can be set.

Syntax: **ATEXT x,y, f, x$**

Where **x** and **Y** coordinates for the upper right corner of the text (the GfA **TEXT** command uses the lower right corner), **F** is the font (0,1 or 2) and **X$** is the string of characters.

Substitutions: Older computer don't offer graphical text, which you can create manually using graphics commands like **PSET** or **PRESET** to draw ornate letters and then created "blitters" using **PUT** and **GET** or **SSHAPE** and **GSHAPE** functions.

Hi Soft Professional BASIC can use the library calls **v_gtext** to position the text and **vst_font** to set the current font .

SUBSTITUTION FACTOR: Fair to Excellent ` **PORTABILITY**: None

ATN or **ATN()**

Found on virtually all subsets. Not found on Apple Integer and TRS-80 Level I BASICs. You can't use this call under FutureBASIC's "Mini-runtime" package.

This function returns the "Arc tangent" angle in radians of the "Tangent" angle value in radians of X.

Syntax is: A = **ATN(X)**

Timex/Sinclair/Spectrum syntax is: A = **ATN** X

Uses include programs which require that visual wave forms be created on the computer screen (music or sound editing programs) or for finding the angle of an object when you know its distance from you and its height by using the formula: ANGLE = **ATN**(Height/Distance)

You should have knowledge of advanced math, such as geometry, trigonometry and calculus to be fluent in all the uses of **ATN.**

To convert degrees to radians use the following formulas:

Radians = Degrees * (3.1416/180)
or
Radians = Degrees * (PI/180)

SUBSTITUTION FACTOR: None **PORTABILITY:** Very Good to Excellent

ATTR()

(Timex/Sinclair Spectrum BASIC)

Returns the color value of any pixel in and all states (**PAPER, INK, BRIGHT, FLASH**).

Syntax: A = **ATTR(x,y)**
PRINT ATTR(x,y)
IF ATTR(x,y) = X **THEN**...

The value returned or displayed is then checked against these parameters:

If the value is greater than 127, then **FLASH** is on (128 is added to the total of the other parameters), otherwise **FLASH** is off (nothing is added).

If the value is greater than 63 then **BRIGHT** is on (64 is added to the value) if the value is 63 or less then **BRIGHT** is off (nothing is added).

The **PAPER** color is determined by dividing the remaining value by 8 and using the whole number. Then, multiply 8 * this whole number and subtract the remainder to get your **INK** color.

For example: You get a value of 200 from **ATTR**. Subtract 128 from 200. This leaves you with 72. Subtract 64 from 72. This leaves 8. Then 8 divided by 8 is 1 and leaves 0. So **FLASH** is on, **BRIGHT** is on, **PAPER** color is 1 and **INK** is 0.

If you get a value of 232. Subtract 128 (leaving 104), subtract 64 (leaving 40), divide by 8 (5 leaving nothing). **FLASH** and **BRIGHT** are on, **PAPER** is 5, **INK** is 0.

If you get 51 then you know that both **BRIGHT** and **FLASH** are off (it is under both 128 and 64). Then you divided 51 by 8 (which give you 6 and a fraction). Multiply 6 * 8 (which gives you 48). Subtract 48 from 51 (giving you 3). **PAPER** is 6, **INK** is 3.

Here is a small program that may help you:

```
10 PRINT "Enter X coordinate"
20 INPUT X
30 PRINT "Enter Y coordinate"
40 INPUT Y
50 Z=ATTR(X,Y)
60 PRINT "Attribute value is:   ";Z
70 IF Z > 127 THEN F=1
72 IF F=0 THEN ZA=Z
74 IF F=0  GOTO 90
80 IF ZA=Z-128
90 IF ZA > 63 THEN B=1
92 IF B=0 THEN ZB=ZA
94 IF B=0 GOTO 90
100 ZB=ZA-64
110 ZC=ZB/8
120 PC=INT ZC
130 ZD=PC*8
140 IC=ZB-ZD
150 IF F=1 PRINT "Flash on"
160 IF F=0 PRINT "Flash off"
170 IF B=1 PRINT "Bright on"
180 IF B=0 PRINT "Bright off"
190 PRINT "Paper color is #:   ";PC
200 PRINT "Ink color is #:   ";IC
210 END
```

If the **ATTR** can't be assigned to Z then replace line 50 with **PRINT** "Enter the number below" and add a new line: 65 **INPUT** Z

In order for this program to work, your background (**PAPER**) should have a special color value for you to know and then be able to find. It would also help if you could arrange for the pixel checked to be where a letter ends up as the program runs or to place a letter or character in an area you're checking. Remember, the screen will scroll when you press the <**RETURN**> key after an input.

Substitutions: On most modern subsets of BASIC, including IBM PC, Atari ST, Amiga and Macintosh subsets, **POINT(x,y)** returns the color value of the pixel. Atari ST BASIC 2.0 uses **ASK RGB.** True BASIC can use **ASK COLOR** and **ASK COLOR MIX.** Visual BASIC can use **RGB().** Future BASIC can use **FN GETCOLOR(**). Quick BASIC Macintosh uses **TOOLBOX** routine **GETCOLOR**. Apple IIc uses **SCRN.**

SUBSTITUTION FACTOR: Fair to Good **PORTABILITY:** None

ATTRIB

(Turbo/Power BASIC)

Allows you to set the IBM PC DOS file attribute of a designated file.

Syntax: **ATTRIB "filename", a**

Where "filename" is any file in the current default directory or any specified DOS parameter (e.g. "D:\ACCTS\ACCT.DAT") and "a" is either 0 (normal), 1 (read only), 2 (hidden), 4 (system) or 32 (archive).

Substitutions: Quick BASIC can use a DOS interrupt to achieve the same function , see *PC INTERRUPT* section in the appendix.Hi Soft Professional Atari ST can use the library call **FNfattrib%** to change the same parameters. GfA Atari 3.0+ can use **FSETDTA** to change the file attribute bit. Most other BASICs have no direct provision for setting a file's attributes.

SUBSTITUTION FACTOR: None to Poor **PORTABILITY:** None

AUTOCLIP

(FutureBASIC for Macintosh)

Adjusts the status of the window clipping protection (default is **AUTOCLIP = _true**) and affects drawing speed. Must be used with care, as you can overdraw or re-write areas of all windows and you may not wish this to occur. All windows are affected.

Syntax: **AUTOCLIP = _false**
AUTOCLIP = _true

Test before using (Future BASIC version 1.01 did not behave as documented using a Pocketbook 180 -- no difference was seen between _true and _false in our tests).

AUTOCLIP = _false turns off the over write protection and tends to speed drawing, however you can damage the appearance of a window or erase parts of the controls accidentally.

This is an option to be used by programmers who know why they are defeating the clipping routines (which normally prevent writing over buttons and controls).

Substitutions: Generally, on the Atari ST, Amiga and Macintosh the "work place" or "work area" is always inside the window and coordinates are referenced to the window. Clipping is handled by the operating system. You generally don't want to work outside these automatically clipped areas. Hi Soft BASIC for Amiga uses WINDOWS parameters for dealing with clipping (Hi Soft PRO uses **vs_clip** library routine, GfA uses **ACLIP** and **CLIP**.)

There are no direct substitutions. Using certain drawing primitives or **PUT** and **GET** commands under other BASICs for the Atari ST, Amiga and Macintosh BASIC can possibly overdraw windows accidentally. You simply have to keep track of your coordinates and create routines to create clipping areas if you find it necessary.

On the IBM PC under some BASICs you can create a view port or world coordinate system and make areas of the screen automatically clip. See **VIEW** and **WINDOW** for the IBM PC and compatibles. VB uses **AutoRedraw** and **ClipControls**.

REM out this line on most other BASICs. Also see the *STANDARD INTERFACE AND GUI* section.

SUBSTITUTION FACTOR: None to Fair (manual algorithm required) **PORTABILITY:** None

AutoRedraw and ***AutoSize***

(Visual BASIC for Windows)

A run-time setting that is either set to FALSE or TRUE when you create the program. When **AutoRedraw** is set to **TRUE** it saves all **PRINT** and graphics held in windows to memory (thus consuming memory) so that windows can be re drawn with ease when they are sized or hidden behind other windows. When it is set to **FALSE** more system memory is free for other functions, but redrawing becomes a manual chore. When **AutoSize** is set to **TRUE** images are pre-sized to fit current windows.

Substitutions: This must be done manually on other BASICs by creating a routine that redraws your printed data (from individual variables or an array list) or by using **BLOAD** or **PUT** graphics commands to placed an image into the screen area designated.