Language Reference

A

Aboutmenu(

The AboutMenu function adds an 'About' menu entry to the 'Apple' Menu and returns the Menu Item id of that menu entry.

Syntax

ABOUTMENU(<string expression>)

Example

MenuItem=AboutMenu("About My Program")

Abs(

The ABS(function returns the absolute value of a number.

Syntax

ABS(<numeric expression>)

Example

```
Print ABS(9)        --prints 9
Print ABS(-4.3)      --prints 4.3
```

Add

The ADD command performs the addition function on two integer operators.

Syntax

ADD <integer variable name>,<numeric expression>

Example

```
ADD Myvar,10      -- Adds 10 to variable Myvar
ADD A,B               -- Adds variable B to A
```

Addmenu(

The Addmenu function adds a menu to the existing Menu Bar and returns the Menu id of that menu.

Syntax

ADDMENU(<string expression>)

Example

FileMenu=AddMenu("File")

Addmenuitem(

The Addmenuitem function adds a menu item to an existing menu and returns the Menu Item id of that menu item.

Syntax

ADDMENUITEM(<integer expression>,<string expression>,<string expression>)

Example

MenuItemId=AddMenuItem(FileMenu,"Quit","Q")

Note
The first parameter is the Menu id of the Menu to which the Menu Item should be appended.
The second parameter is the string which should appear on the Menu Item when it is displayed.
The third parameter is the keyboard equivalent character. If a null string is supplied in this parameter then no keyboard equivalent will appear in the menu item.

Alen(

The ALEN(function returns the length (number of elements) in the given array.

Syntax

ALEN(<array variable name>)

Example

```
Global Myvar As Integer(10)
Print ALEN(Myvar)       -- prints 10.
```

Note

Arrays are dynamically resizable. Use this function to determine the

current size. See REDIM and ARRAYLOAD.

And

The AND operator performs a logical AND on two numeric values or conditional
expressions.

Syntax
<numerical or logical expression> AND <numerical or logical expression>

Example
A AND B -- returns the bitwise AND of A and B
IF A>B AND C<D -- performs a logical AND on the two conditional
expressions.

Arrayload

The ARRAYLOAD command loads an array into memory from a DATA type resource
and dynamically resizes it.

Syntax
ARRAYLOAD <array variable name>,<DATA resource id>

Example
Global Myarray As Integer(0)
ARRAYLOAD Myarray,128

Note
The ALEN(function can be used to determine the size of the array.

Arraysave

The ARRAYSAVE command saves the contents of an array in a DATA type resource.

Syntax
ARRAYSAVE <array variable name>,<DATA resource id>

Example
ARRAYSAVE Myarray,128

Note
The ARRAYSAVE and ARRAYLOAD commands are useful for saving data
between invocations of a program run.

Asc(

The ASC function returns the ASCII value of the first character in a string.

Syntax
ASC(<string expression>)

Example
```
Print ASC("Chalfont")        -- prints 67.
```

Ascan(

The ASCAN function searches a single dimensional string array for a chosen string and returns the index position of that string, or zero if that string is not contained within the array.

Syntax
ASCAN(<string array variable>,<string expression>,<integer expression>)

Example
```
Mystr(1)="ABC"
Mystr(2)="DEF"
Mystr(3)="GHI"
Idx=ASCAN(Mystr,"DEF",0)        -- idx is set to 2
```

Note
If the <integer expression> is set to zero then the whole array is searched.
If set to any number less than that then only part of the array is searched.

B

Break

Forces execution, within a DO CASE construct to continue execution after the ENDCASE statement.

Syntax

BREAK

Example

See DO CASE.

Byref(

The Byref(function passes the address of a variable to a procedure or function so that its value may be altered therein. The default is to pass parameters by value.

Syntax

BYREF(<variable name>)

Example

Retval=Myfunc(A, BYREF(B))

Byte(

The BYTE(function performs type conversion into BYTE datatype format.

Syntax

BYTE(<expression>)

Example

B=BYTE(I)

Note

If the <expression> yields an Integer or Word result then it is truncated to 8 bits by the BYTE(function.

If the <expression> yields a Char, Structure, Str255 or String result then the result will be equivalent to the ascii value of the first character.

The Byte function is not permitted on a Float expression.

C

Case

See DO CASE

Char(

The CHAR(function performs type conversion into CHAR datatype format.

Syntax

CHAR(<expression>)

Example

C=CHAR(I)

Note

If the <expression> yields a Byte result then this will result in a Char datatype containing a single character equivalent to the ascii character contained in the Byte field.
If the <expression> yields a Word or Integer result then this will be converted into a character string containing the value of the Word or Integer.
If the <expression> yields a Str255 or String result then this will be converted into the equivalent Char representation.
The datatypes Char and Structure are equivalent.

Chr(

The CHR(function converts an Integer datatype into a single character string containing the ascii representation of the lower 8 bits of the Integer.

Syntax

CHR(<integer expression>)

Example

C=CHR(I)

Close

Closes a disk file.

Syntax

CLOSE #<file number>

Example

```
OPEN "Myfile" AS #1
------
CLOSE #1
```

Close Console

Closes the console window

Syntax

```
CLOSE CONSOLE
```

Cls

The CLS command clears the console screen

Syntax

```
CLS
```

D

Debugger

The DEBUGGER command enters a machine code debugger such as Macsbug if installed.

Syntax

```
DEBUGGER
```

Dec

The DEC command decrements the value of an integer variable. This is equivalent to but much faster than var=var-1.

Syntax

```
DEC <integer variable name>
```

Example

```
DEC A
```

Default

The default command forms part of the Do Case construct and marks the default statement set to be executed. Equivalent to CASE -1.

Syntax
DEFAULT

Example
See DO CASE.

Disablemenu

The Diablemenu command sets the Enabled status of a Menu or Menu Item to Off thus dimming the entry. Once disabled, the user can not select the menu entry.

Syntax
DISABLEMENU <integer expression>,<integer expression>)

Example
DisableMenu FileMenu,2

Note
The first parameter specifies the Menu id of the menu in question.
The second parameter specifies the Menu Item id of the item to be disabled.
If this parameter is set to zero, the whole menu is disabled.

Do

The Do command invokes a procedure optionally passing parameters.

Syntax
DO <procedure name>[(parm,parm....)]

Example
DO Myproc(Var1, Byref(Var2))

Do Case

The Do Case command marks the start of a CASE construct.

Syntax
DO CASE

Example
```
DO CASE
    CASE A=1
        Do Myproc(B)
```

```
        BREAK
    CASE (A,5) AND (B=2)
        C=Myfunc(A)
        BREAK
    DEFAULT
        C=0
ENDCASE
```

Dpeek(

The Dpeek (Double Peek) function returns the 16 bit value stored in a given memory location.

Syntax

DPEEK(<integer value>)

Example

Myword=DPEEK(10538)

Dpoke

The Dpoke command places a 16 bit value into a given memory location. Use with care.

Syntax

DPOKE ,<integer memory address>,<integer value>

Example

DPOKE 10538,A

E

Else

The Else statement forms part of the IF,ELSE,ENDIF construct.

Syntax

ELSE

Example

See IF.

Enablemenu

The Enablemenu command sets the Enabled status of a Menu or Menu Item to On thus highlighting the entry. Once enabled, the user can select the menu entry.

Syntax
ENABLEMENU <integer expression>,<integer expression>)

Example
EnableMenu FileMenu,2

Note
The first parameter specifies the Menu id of the menu in question.
The second parameter specifies the Menu Item id of the item to be enabled. If this parameter is set to zero, the menu is enabled and the menu items maintain their previous status.

End

The END statement returns program control to the Finder.

Syntax
END

Endcase

The ENDCASE statement terminates a DO CASE construct.

Syntax
ENDCASE

Example
See DO CASE

Endif

The Endif statement terminates an IF,ELSE,ENDIF construct.

Syntax
ENDIF

Example
See IF

Endstruct

The Endstruct statement terminates a STRUCTURE, ENDSTRUCT declaration.

Syntax

```
ENDSTRUCT
```

Example

See Structure.

Eof(

The EOF(function is used to determine if the file pointer of a file has reached the End Of File.

Syntax

```
EOF(<file_number>)
```

Example

```
REPEAT
    INPUT #1,Mystr,Myint
    Do Myproc(Mystr,Myint)
UNTIL EOF(1)
```

Err

Err is a reserved variable which contains the code number of the last error which occurred. Where Err is set to zero, this indicates that no error has occurred.

Syntax

```
ERR
```

Example

```
IF Err <>0
    Print "An error has occurred"
Endif
```

Note

A negative error number denotes an operating system error and a positive error number denotes a run time error within your Basic program.

F

False

False, when used in an expression yields a result of zero and is used as a documentation aid.

Syntax

FALSE

Example

If A=False

Float(

The Float function performs type conversion into FLOAT datatype format.

Syntax

FLOAT(<expression>)

Example

MyFloat=FLOAT(Myint)

Note

If the <expression> yields an Word or Integer result then it is converted
to a Float value by the FLOAT(function.
If the <expression> yields a Char, Str255 or String result then the result will be equivalent to the numeric value of the string.

For

The FOR statement creates a loop that executes a given number of times.

Syntax

FOR <numeric variable> = <numeric expression> TO
<numeric expression> [STEP <numeric expression>]

Example

```
FOR I=1 TO 10
    PRINT I
NEXT I
```

will print the numbers 1 through to 10 on the console.

For J=10 TO 1 STEP -2

```
        PRINT J
    NEXT J
```

will print the numbers 10,8,6,4,2 on the console.

FSOpen

FSOpen opens a disk file using a System 7 filespec.

Syntax

FSOPEN <fsspec structure> [FOR <mode>] AS [#]<file number> [LEN=<record length>]

Example

```
Procedure Myproc()
Local Reply As Structure
    Local sfGood As Byte
    Local sfReplacing As Byte
    Local sfType As Integer
    Local sfFile As Char[70]
    Local sfScript As Word
    Local sfFlags As Word
    Local sfIsFolder As Byte
    Local sfIsVolume As Byte
    Local sfReserved1 As Integer
    Local sfReserved2 As Word
Endstruct

Local FSSpec As Structure
    Local vRefNum As Word
    Local parID As Integer
    Local name As Str255[63]
Endstruct

    _StandardGetFile(0,Word(1),Char("TEXT"),Reply)
    If Integer(sfGood)<>0
        FSSpec=sfFile
        FSOpen FSSpec For Input As #1
    Endif

    _StandardPutFile(Str255("Prompt"),Str255("Default name"),Reply)
    If Integer(sfGood)<>0
        FSSpec=sfFile
        FSOpen FSSpec For Output As #1
```

```
    Endif
```

In the above example, the toolbox trap _StandardGetFile is used to prompt the user for a file to open and to return a file spec. The FSOpen command is then used to open the file.
The _StandardPutFile is then used to prompt the user for a file to save. The FSOpen command is then used to open the file for Output.
See 'Inside Macintosh Files' for more details on StandardGetFile and StandardPutFile.
See OPEN for permissible modes etc.

Function

The Function statement declares the start of a user defined function.

Syntax

FUNCTION <name>(<parameter list>) RETURNING <data type>

Example

FUNCTION Myfunc(A,B,C) RETURNING INTEGER

Declares the start of a function, Myfunc, which receives 3 parameters (defined by
the PARAMETER statement) and returns an Integer data type to its caller.

G

Get

This statement retrieves a record from a files which has been opened in RANDOM mode.

Syntax

GET #<file_number>,<record_number>,<variable name>

Example

GET #1,123,Mystruct

Note
The variable to which the data is read may be a STRING or a STRUCTURE type.

Getappmessage

This function returns the command with which the application has been started.

Syntax
GETAPPMESSAGE()

Example
Cmd=GETAPPMESSAGE() .. see Getappfile

Note
GETAPPMESSAGE return 0 if the command is to Open file(s) and 1 if the
command is to Print the file(s).

Getappcount

This function returns the number of files with which the application has been started.
(ie the number of files which have been dragged onto the application, or one if a file has been double clicked).

Syntax
GETAPPCOUNT()

Example
Mycount=GETAPPCOUNT() .. see getappfile

Getappfile

This function returns into a structure the details of a file to be opened or printed.

Syntax
GETAPPFILE(<integer expression>)

Example
```
Global I As Integer
Global Command As Integer

Global AppStruct As Structure
    Global AvRefNum As Word
```

```
    Global Atype As Integer
    Global AversNum As Byte
    Global Afiller As Byte
    Global AfileName As Str255 [64]
Endstruct

Command=GetAppMessage()

If GetAppCount()>0
    For I=1 To GetAppCount()
        AppStruct=GetAppFile(I)
        Do Case
            Case Command=0   'Open file
                Do MyOpenFile(Integer(AvRefNum),Atype,String(AfileName)
            Case Command=1 ' Print file
                Do MyPrintFile(Integer(AvRefNum),Atype,String(AfileName)
        Endcase
   Next I
Endif
End
```

Global

The Global statement declares Global variables.

Syntax

```
        GLOBAL <variable name> AS <variable type> [<string length>]
[<dimensions>]
```

Example

```
        GLOBAL I AS INTEGER                 .. declares variable I as an Integer
        GLOBAL ST As STRING [10]        .. declares variable ST as a String of
maximum
                                                                    length, 10
characters.
        GLOBAL A AS INTEGER(20)         .. declares variable A as an Integer
array of
                                                                    20 elements.

        GLOBAL B AS STRING [12] (15) .. declares B as a string array,
maximum 12
                                                                     characters with 15
elements.

        GLOBAL T AS INTEGER(10,30)      .. declares T as an Integer array
```

with 10 rows and 30 columns.

Note
Global variables may only be declared in the 'Initial' module.