ToolType

COLLABORATORS			
	TITLE : ToolType		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 19, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ToolType

1.1 ToolType V1.00

```
Pure Basic - ToolType V1.00

  The 'Tooltypes' are located in a workbench icon. It's some
  strings you can easely write and read to store, for example,
  your program configuration. This is very useful because you
  don't need to create a separate file and the user can modify
  it himslef via the menu 'Icon/Information' of the Workbench
  screen.

Commands summary:

   FreeToolType
   GetNextToolTypeString
   GetNumberOfToolTypes
   GetToolTypeValue
   InitToolType
   MatchToolType
   MatchToolTypeString
   ReadToolTypeDiskInfo
   WriteToolTypeDiskInfo

   ToolType Demo
```

1.2 freetooltype

```
  SYNTAX
FreeToolType(#Obj.w)

  STATEMENT
This statement frees a ToolType Object that was initialized with
ReadToolTypeDiskInfo(), but it doesn't care if the Object to free
isn't initialized.

#Obj
The Object to free.
```

1.3 getnexttooltypestring

```
  SYNTAX
String$ = GetNextToolTypeString(#Obj.w)

  FUNCTION
This function returns the next ToolType and all associated values
as a string.

After the last ToolType is reached the next ToolType that will be
returned is the first one. To find out how many ToolType strings
exist, use the Returnvalue from the function GetNumberOfToolTypes().

A ToolType is in length restricted to 128 bytes by the OS.

#Obj
Object to use.

String
The returned string holds the ToolType and all associated values like
this  "AMIGA=1200|4000".
```

1.4 getnumberoftooltypes

```
  SYNTAX
Result.w = GetNumberOfToolTypes(#Obj.w)

  FUNCTION
This function returns the number of ToolTypes that are present
in the icon's .info file, this should be used to easily fetch
the right number of ToolType strings.

#Obj
Object to use.

Result
Any positive value is the number of ToolTypes there is in the
icon's .info file, but if it's instead  -1  this indicate that
no ToolType at all is present in icon's .info file.
```

1.5 gettooltypevalue

```
  SYNTAX
String$ = GetToolTypeValue(#Obj.w,ToolName$)

  FUNCTION
This function returns the associated values, to the specified ToolType,
as a string.

#Obj
Object to use.
```

```
String
The returned string holds the Tool value like this "1200|4000", but
if the ToolType was not present in icon's .info file the string
will be empty.
```

1.6 inittooltype

```
  SYNTAX
Result.l = InitToolType(Objects.l,*WBStartup)

  FUNCTION
This function set up some basic stuff to handle ToolTypes.

This is the Init routine and should always be called before any
other ToolType function. At the moment it could only be called
once, so if there is a failure with this call the program should
always quit.

Objects
This parameter specifies how many ToolType Objects are needed.
Each Object can hold data about one .info file.

Maximum number of Objects is 2046.

*WBStartup
This is a pointer to a WBStartupMessage, this Message is passed to
the program from WorkBench, *WBStartup is the Returnvalue from the
function WBStartup() that catches the Message.

With this approach Object 0 is always automatically initialized from
the .info file of the double clicked icon and the current directory
is also set to the directory of this icon.

If this parameter isn't a pointer, but instead zero, none of the above
actions have taken place.

Result
If some error has occurred then result is FALSE and there is no point
in calling any other ToolType functions.  Should Always Be Ckecked.
```

1.7 matchtooltype

```
  SYNTAX
Result.w = MatchToolType(#Obj.w,ToolName$,Value$)

  FUNCTION
This function checks if ToolType is present and if it's set to
the specified value.

None of the ToolTypes need to be read into Pure Basic strings for
this function, it's convenient when ToolTypes only need to be
examined.
```

```
ToolName
This is the ToolType to look out for.

Value
This is the value to look out for, could be a zerostring.

Result
If this is -1 the ToolType is not present in icon's .info file, if
it's instead 0 the ToolType is found but it's not set to the specified
value and finally if it's 1 then the ToolType is found and have one
value that is the same as third parameter.
```

1.8 matchtooltypestring

```
  SYNTAX
Result.w = MatchToolTypeString(String$,ToolName$,Value$)

  FUNCTION
This function checks if a Pure Basic string holds a specific
ToolType and a matching value.

String
This is the string to examine.

ToolName
This is the ToolType to check for.

Value
This is the value to check for, could be a zerostring.

Result
If this is -1 the ToolType is not present in Pure Basic string, if
it's instead 0 the ToolType is found but it's not set to the specified
value and finally if it's 1 then the ToolType is found and have one
value that is the same as third parameter.
```

1.9 readtooltypediskinfo

```
  SYNTAX
Result.l = ReadToolTypeDiskInfo(#Obj.w,IconName$)

  FUNCTION
This function reads the icon's .info file that is specified by name
and initializes the ToolType Object.

If the Object is already initialized this function doesn't care and
just creates a new Object without deleting the old one - then
there is no possibile way to preform any actions on the old Object.

Maximum number of ToolTypes that could be handled is 32767,
restricted by this Pure Basic Lib.
```

```
#Obj
Object to use.

IconName
The icon's name: ".info" is automatically added to the end of name.

Result
If this is set to FALSE the icon's .info file couldn't be read.
```

1.10 writetooltypediskinfo

```
  SYNTAX
Result.w = WriteToolTypeDiskInfo(#Obj.w,Array(),IconName$)

  FUNCTION
This function writes the icon's .info file back to disk.

#Obj
Object to use.

Array()
This array hold the strings that contain the new ToolTypes
to be written.

The new ToolTypes have priority over old ToolTypes, so if only
one ToolType is added then all ToolTypes must first be read
into Pure Basic strings with the function GetNextToolTypeString().

Maximum number of ToolTypes that could be handled is 32767,
restricted by this Pure Basic Lib.

IconName
The icon's name: ".info" is automatically added to the end of name.

Result
If this is TRUE then the writing was succesful.
```