ToolType

COLLABORATORS			
	TITLE : ToolType		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 23, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ToolType

1.1 ToolType

```
Pure Basic - ToolType V1.01

  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(#Info.w)

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

#Info
The info to free.
```

1.3 getnexttooltypestring

```
  SYNTAX
String$ = GetNextToolTypeString(#Info.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 return value from the function GetNumberOfToolTypes().

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

#Info
The info 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(#Info.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.

#Info
The info to use.

Result
Any value higher than zero is the number of ToolTypes there is in
the  icon's .info file, but if it's instead  0  this indicate that
no ToolType at all is present in icon's .info file.
```

1.5 gettooltypevalue

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

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

#Info
The info 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(#Infos.l)

  FUNCTION
This is the init routine and should always be called before any
other ToolType function. If there is a failure with this call
then no other function should be called.

#Infos
This parameter specifies how many ToolType infos are needed.
Each info can hold data about one .info file.

Maximum number of infos is 2046.

Special Note:
The second parameter, used in version 1.00, that could hold
a value of zero or be a return value from WBStartup() is gone.
This is now done automatically with the restriction that
WBStartup() must be called befor InitToolType().

If InitToolType() detects that the program is started from
WorkBench then #Info 0 will be automatically initialized from
the .info file of the double clicked icon and the current
directory is also set to the directory of this icon.

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

1.7 matchtooltype

```
  SYNTAX
Result.w = MatchToolType(#Info.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.

#Info
The info to use.
```

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

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

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 empty string.

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(#Info.w,IconName$)

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

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

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

```
#Info
The info 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(#Info.w,Array(),IconName$)

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

#Info
The info 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 or changed then all ToolTypes
must first be read into Pure Basic strings with the function
GetNextToolTypeString() and then written back.

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 successful.
```