Misc

COLLABORATORS

	TITLE : Misc	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Misc

1.1 Misc V1.30

```
PureBasic Misc library V1.30

  This library provides a lot of very useful but inclassifiable
  functions..

Commands summary:

   Delay
   FPU
   GetCLIArg
   GetFilePart
   GetPathPart
   MMU
   MouseButtons
   MouseWait
   NumberOfCLIArgs
   PeekB
   PeekL
   PeekS
   PeekW
   PokeB
   PokeL
   PokeS
   PokeW
   Print
   PrintN
   PrintNumber
   PrintNumberN
   Processor
   ProgramPriority
   RunProgram
   VWait
```

1.2 delay

```
  SYNTAX
Delay(Time)

  FUNCTION
Halts the program execution for the given time.

Time: Number of seconds to wait.
```

1.3 fpu

```
  SYNTAX
Result = FPU()

  FUNCTION
Return if a FPU (Floating Point Unit) is present on the Amiga.

Possible values:

0: No FPU available
1: 68881 FPU detected
2: 68882 FPU detected
```

1.4 getcliarg

```
  SYNTAX
Argument$ = GetCLIArg(Position.w)

  FUNCTION
Returns the argument at position (Position.w) that was passed to the
program at CLI start. If less arguments were given the function
returns NULL.
It is assumed that the function  NumberOfCLIArgs  has been called
once before for this function to work correctly.

Example:

  Second$ = GetCLIArg(2)

Second$ will be 'b', if the program 'TestProg' has been startet with
> TestProg "a c" b
from CLI.
```

1.5 getfilepart

```
  SYNTAX
FileName$ = GetFilePart(String$)

  FUNCTION
Returns the FileName of a path and filename string.
```

```
Example:

  FileName$ = GetFilePart("Dh0:Games/SuperFrog/SuperFrog.exe")

FileName$ will be 'SuperFrog.exe'.
```

1.6 getpathpart

```
  SYNTAX
PathName$ = GetPathPart(String$)

  FUNCTION
Returns the full path of a path and filename string.

Example:

  PathName$ = GetPathPart("Dh0:Games/SuperFrog/SuperFrog.exe")

PathName$ will be 'Dh0:Games/SuperFrog/'.
```

1.7 mousebuttons

```
  SYNTAX
PressedButtons.w = MouseButtons()

  FUNCTION
Returns the currently pressed mousebuttons (1 = left, 2 = right, 3 = both,
0 = none).
```

1.8 mmu

```
  SYNTAX
Result = MMU()

  FUNCTION
If no MMU (Memory Management Unit) is present on the Amiga, the command
returns 0, else a MMU is present.
```

1.9 mousewait

```
  SYNTAX
MouseWait()

  FUNCTION
Waits for a left mouse button click.
```

1.10 numberofcliargs

```
  SYNTAX
NumOfArgs.w = NumberOfCLIArgs()

  FUNCTION
Returns the number of arguments that were passed to the program at CLI
start.
The single arguments must be separated by spaces. If an argument contains
spaces it must be surrounded by double quotes (Chr(34)). If the number of
double quotes is odd the function returns 0, independent of the argument
string's contents.
```

1.11 peekx

```
  SYNTAX
Result = PeekB/W/L/S(*Address)

  FUNCTION
Return a byte, word, long or string, depending of the suffix used, located
at the given *Address.
```

1.12 pokex

```
  SYNTAX
PokeB/W/L/S(*Address, Data)

  FUNCTION
Put the given data to the specified *Address. It can put a byte, word,
long or string depending of the the suffix used.
```

1.13 print

```
  SYNTAX
Print(String$)

  FUNCTION
Prints the given 'String$' to the default output (CLI/Shell).
```

1.14 printn

```
  SYNTAX
PrintN(String$)

  FUNCTION
Prints the given 'String$' to the default output (CLI/Shell) and adds
an 'end of line' character at the end of the string.
```

1.15 printnumber

```
  SYNTAX
PrintNumber(Number)

  FUNCTION
Prints the given number to the default output (CLI/Shell).
```

1.16 printnumbern

```
  SYNTAX
PrintNumberN(Number)

  FUNCTION
Prints the given number to the default output (CLI/Shell) and adds
a 'end of line' character.
```

1.17 vwait

```
  SYNTAX
VWait()

  FUNCTION
Waits until the next frame begins. It's also known as Vertical Blank. Used
to synchronize animation with the display.
```

1.18 runprogram

```
  SYNTAX
RunProgram(DefaultPath$, CommandLine$, ASynchrone, Stack)

  FUNCTION
Launches an external program from your program, using the given
parameters.

DefaultPath$: Full path to be used by default for the launched program.

CommandLine$: Command to execute (Path and name of the program to launch).

ASynchrone: If set to 1, will launch the program in asynchrone matter, so
            your program will continue immediatly after the launch. Else
            your program will be halted until the launched program quits.

Stack: Stack value for the launched program. Set it at least to 4096 if you
       don't know what it does !
```

1.19 processor

```
  SYNTAX
Result = Processor()

  FUNCTION
Return which processor is present in the Amiga.

Possible values:

0: 68000
1: 68010
2: 68020
3: 68030
4: 68040
6: 68060
```

1.20 programpriority

```
  SYNTAX
OldPriority.b = ProgramPriority(NewPriority)

  COMMAND
It allows the programmer to set the priority of the program. It could
be very useful when launching a task which uses a lot of cpu time
during long periods (ie: rendering, compression...) and shouldn't lock
the whole system. So set a priority of -1 and it will multitask very
well!

In another side, a game needs most system resources, so when doing a fast
arcade game in a multitasking environment, you must set your task priority
to 10 (at least). Don't forget to reduce it when no more action is needed
(ie: menus, waiting...)
```