

**Misc**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> Misc		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		January 23, 2025	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>Misc</b>	<b>1</b>
1.1	Misc V1.10 . . . . .	1
1.2	delay . . . . .	1
1.3	getfilepart . . . . .	1
1.4	getpathpart . . . . .	2
1.5	mousewait . . . . .	2
1.6	peekx . . . . .	2
1.7	pokex . . . . .	2
1.8	print . . . . .	3
1.9	printn . . . . .	3
1.10	printnumber . . . . .	3
1.11	printnumbern . . . . .	3
1.12	programpriority . . . . .	4
1.13	runprogram . . . . .	4
1.14	vwait . . . . .	4

# Chapter 1

## Misc

### 1.1 Misc V1.10

Pure Basic Misc library V1.10

Diese Library bietet eine ganze Menge an sehr nützlichen Funktionen an, die sich jedoch schlecht thematisch sortieren lassen...

Befehlsübersicht:

- Delay
- GetFilePart
- GetPathPart
- MouseWait
- PeekX
- PokeX
- Print
- PrintN
- PrintNumber
- PrintNumberN
- ProgramPriority
- RunProgram
- VWait

### 1.2 delay

SYNTAX  
Delay(Time)

FUNCTION  
Hält die weitere Programmausführung für die angegebene Zeit an.

Time: Wartezeit in Sekunden

### 1.3 getfilepart

---

**SYNTAX**

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

**FUNCTION**

Ermittelt den Dateinamen aus einem kompletten Pfad-/Dateistring.

Beispiel:

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

FileName\$ ergibt 'SuperFrog.exe'.

## 1.4 getpathpart

**SYNTAX**

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

**FUNCTION**

Ermittelt den vollen Pfad aus einem kompletten Pfad-/Dateistring.

Beispiel:

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

PathName\$ ergibt 'Dh0:Games/SuperFrog/'.

## 1.5 mousewait

**SYNTAX**

```
MouseWait ()
```

**FUNCTION**

Wartet auf einen Klick mit der linken Maustaste.

## 1.6 peekx

**SYNTAX**

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

**FUNCTION**

Gibt ein Byte, Word, Long oder String, abhängig von der benutzten Endung, von der angegebenen \*Adresse zurück.

## 1.7 pokex

## SYNTAX

PokeB/W/L/S(\*Address, Data)

## FUNCTION

Schreibt die angegebenen Daten an die vorgegebene \*Adresse. Es kann ein Byte, Word, Long oder String, abhängig von der angegebenen Endung, benutzt werden.

## 1.8 print

## SYNTAX

Print(String\$)

## FUNCTION

Gibt den angegebenen 'String\$' auf der Standardausgabe (CLI/Shell) aus.

## 1.9 printn

## SYNTAX

PrintN(String\$)

## FUNCTION

Gibt den angegebenen 'String\$' auf der Standardausgabe (CLI/Shell) aus und fügt ein 'end of line' Zeichen am Ende des Strings hinzu.

## 1.10 printnumber

## SYNTAX

PrintNumber(Number)

## FUNCTION

Gibt die angegebene Zahl auf der Standardausgabe (CLI/Shell) aus.

## 1.11 printnumbersn

## SYNTAX

PrintNumberN(Number)

## FUNCTION

Gibt die angegebene Zahl auf der Standardausgabe (CLI/Shell) aus und fügt ein Zeilenende-Zeichen ('end of line') hinzu.

---

## 1.12 programpriority

### SYNTAX

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

### COMMAND

Erlaubt dem Programmierer die Priorität des Programms festzulegen. Dies ist sehr nützlich, wenn Sie einen Task, welcher über längere Zeit sehr viel CPU-Rechenzeit beansprucht (z.B. Rendering, Kompression,...), starten und dieser nicht das ganze System lahmlegen soll. Setzen Sie in diesem Fall die Priorität auf -1 und Ihr Programm läuft wunderbar im Multitasking!

Auf der anderen Seite benötigt ein Spiel die meisten Systemressourcen. Wenn Sie ein schnelles Action-Spiel in einer Multitasking-Umgebung laufen lassen, müssen Sie die Task-Priorität auf (mindestens) 10 setzen. Vergessen Sie nicht, diese zu reduzieren, wenn keine "große Action" mehr passiert, z.B. in Menüs, in einer Warteschleife...

## 1.13 runprogram

### SYNTAX

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

### FUNCTION

Startet ein externes Programm mit den angegebenen Parametern aus Ihrem Programm heraus.

**DefaultPath\$:** Voller Dateipfad, den das zu startende Programm als Standardeinstellung nutzen soll.

**CommandLine\$:** auszuführender Befehl (Pfad und Dateiname des zu startenden Programms).

**ASynchrone:** Wenn auf 1 gesetzt, wird das Programm in asynchroner Weise gestartet, sodaß Ihr Programm unmittelbar nach dem Start fortgesetzt wird. Andernfalls wird Ihr Programm angehalten, bis das gestartete Programm wieder beendet wird.

**Stack:** Stack Wert für das gestartete Programm. Setzen Sie diesen Wert mindestens auf 4096, wenn Sie nicht genau wissen, was es macht!

## 1.14 vwait

### SYNTAX

```
VWait()
```

### FUNCTION

Wartet, bis das nächste Frame beginnt. Dies ist auch als Vertical Blank bekannt. Wird benutzt, um Animationen mit der Anzeige zu synchronisieren.