

**BitMap**

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

	<i>TITLE :</i> BitMap		
<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>BitMap</b>	<b>1</b>
1.1	BitMap . . . . .	1
1.2	allocatebitmap . . . . .	1
1.3	allocatelinearbitmap . . . . .	2
1.4	freebitmap . . . . .	2
1.5	initbitmap . . . . .	2
1.6	bitmapid . . . . .	2
1.7	bitmapprastport . . . . .	3
1.8	usebitmap . . . . .	3
1.9	showbackbitmap . . . . .	3
1.10	showbitmap . . . . .	3

# Chapter 1

## BitMap

### 1.1 BitMap

PureBasic - BitMap library

'BitMap' are memory area used to store and later to display pictures or graphical objects. The 'BitMap' is so called a planar display as its a superposition of single 'BitPlanes'. Each 'BitPlanes' contain only 0 and 1 and more there is BitPlanes superposed and more you can have differents colours. It's the 'Depth' of the BitMap. For example, a BitMap of Depth '8' (8 bitplanes superposed) can have up to  $2^8$  colours or 256 colours. Planar is the native Amiga display format.

Commands summary:

```
AllocateBitMap
AllocateLinearBitMap
BitMapID
BitMapRastPort
FreeBitMap
InitBitMap
UseBitMap
ShowBackBitMap
ShowBitMap
```

Example:

```
Double buffering
```

### 1.2 allocatebitmap

SYNTAX

```
BitMapID.l = AllocateBitMap(#BitMap, Width, Height, Depth)
```

FUNCTION

---

Create a new BitMap object with given parameters. If the result is NULL, then there is not enough memory, so STOP your bitmap manipulations !

### 1.3 allocatelinearbitmap

#### SYNTAX

```
BitMapID.l = AllocateLinearBitMap(#BitMap, Width, Height, Depth)
```

#### FUNCTION

Create a new BitMap object with given parameters. This bitmap is bit special, as all the planes are grouped into a single block of memory. This kind of BitMaps should be used only for ChunkyToPlanar conversion. It's not compatible with graphic cards, so please use AllocateBitMap for standard programs.

If the result is NULL, then there is not enough memory, so don't use it !

### 1.4 freebitmap

#### SYNTAX

```
FreeBitMap(#BitMap)
```

#### STATEMENT

Free the given BitMap object and release the previously allocated memory.

### 1.5 initbitmap

#### SYNTAX

```
result.l = InitBitMap(#NumBitMapMax)
```

#### FUNCTION

Init all the BitMap environments for later use. You must put this function at the top of your source code if you want to use the BitMap commands.

#NumBitMapMax : Maximum number of BitMaps to handle.

### 1.6 bitmapid

#### SYNTAX

```
BitMapID.l = BitMapID()
```

#### FUNCTION

Returns the BitMap pointer.

---

## 1.7 bitmaprastport

### SYNTAX

```
RastPort.l = BitMapRastPort()
```

### FUNCTION

Returns the current BitMap's rastport. Needed to use the 2D Drawing functions available in the 2D Drawing library.

## 1.8 usebitmap

### SYNTAX

```
UseBitMap(#BitMap)
```

### STATEMENT

Change the currently used BitMap to #BitMap.

## 1.9 showbackbitmap

### SYNTAX

```
ShowBackBitMap(#BitMap, ScreenID, x, y)
```

### STATEMENT

Display the given bitmap in the back part of a dualplayfield screen at position x, y. If the Screen is not in 'DualPlayfield' mode, don't use this command ! See the 'Screen' library guide for more information about the 'DualPlayField' mode.

If you do a multitask game, don't forget to use the ProgramPriority() function to have a high priority, gaining much more cpu time.

## 1.10 showbitmap

### SYNTAX

```
ShowBitMap(#BitMap, ScreenID, x, y)
```

### STATEMENT

Display the given bitmap on the screen at position x, y. This function is 100% OS friendly and allows very fast double-buffering.

If you do a multitask game, don't forget to use the ProgramPriority() function to have a high priority, gaining much more cpu time.

---