

**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	bitmapid . . . . .	2
1.5	bitmapprastport . . . . .	2
1.6	freebitmap . . . . .	2
1.7	initbitmap . . . . .	3
1.8	usebitmap . . . . .	3
1.9	showbackbitmap . . . . .	3
1.10	showbitmap . . . . .	3

# Chapter 1

## BitMap

### 1.1 BitMap

Pure Basic - BitMap Library

Eine 'Bitmap' ist ein Speicherbereich für das Speichern und spätere Anzeigen von Bildern oder grafischen Objekten. Die 'Bitmap' wird auch "planares Display" genannt, da sie das Ergebnis einzelner 'Bitplanes' (Ebenen) ist. Jede Ebene beinhaltet nur 0 oder 1 und je mehr Ebenen (Bitplanes) übereinanderliegen, desto mehr Farben sind möglich. Dies ist auch die sogenannte 'Tiefe' der Bitmap. Zum Beispiel kann eine Bitmap mit der Tiefe '8' (8 kombinierte Bitplanes) bis zu  $2^8$  bzw. 256 Farben haben. 'Planar' ist das standardmäßige Darstellungsformat auf dem Amiga.

Befehlsübersicht:

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

Beispiel:

```
Double Buffering
```

### 1.2 allocatebitmap

SYNTAX

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

FUNCTION

---

Erstellt ein neues Bitmap Objekt mit den angegebenen Parametern. Ist das Resultat NULL, dann ist nicht genug Speicherplatz vorhanden. Stoppen Sie Ihre weiteren Bitmap Manipulationen!

### 1.3 allocatelinearbitmap

#### SYNTAX

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

#### FUNCTION

Erstellt ein neues Bitmap Objekt mit den angegebenen Parametern. Diese BitMap ist ein wenig speziell, da sich alle Ebenen (Planes) in einem einzelnen Speicherblock befinden. Diese Art von BitMaps sollte nur für ChunkyToPlanar Umwandlungen benutzt werden. Sie sind nicht kompatibel mit Grafikkarten, benutzen Sie also bitte AllocateBitMap für Standard-Programme.

Ist das Resultat NULL, dann ist nicht genug Speicher vorhanden, benutzen Sie die BitMaps dann nicht!

### 1.4 bitmapid

#### SYNTAX

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

#### FUNCTION

Gibt den Zeiger (Pointer) der Bitmap zurück.

### 1.5 bitmaprastport

#### SYNTAX

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

#### FUNCTION

Gibt den Rastport der aktuellen Bitmap zurück. Wird für die 2D Zeichen-Befehle der 2D Library benötigt.

### 1.6 freebitmap

#### SYNTAX

```
FreeBitMap(#BitMap)
```

#### STATEMENT

Gibt das angegebene Bitmap Objekt und den zuvor reservierten Speicherplatz frei.

---

## 1.7 initbitmap

### SYNTAX

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

### FUNCTION

Bereitet die gesamte Bitmap Umgebung zum späteren Einsatz vor. Sie müssen diese Funktion am Anfang Ihres Programmcodes aufrufen, wenn Sie Bitmap Befehle nutzen möchten.

#NumBitMapMax : maximale Anzahl Bitmaps in Ihrem Programm

## 1.8 usebitmap

### SYNTAX

```
UseBitMap(#BitMap)
```

### STATEMENT

Ändert die aktuell benutzte Bitmap auf #BitMap.

## 1.9 showbackbitmap

### SYNTAX

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

### STATEMENT

Zeigt die angegebene BitMap in hinteren Teil des DualPlayField Bildschirms an der Position x, y an. Wenn sich der Bildschirm nicht im DualPlayField - Modus befindet, verwenden Sie nicht diesen Befehl! Sehen Sie im Kapitel zur 'Screen' Library nach, um weitere Informationen zum DualPlayField Modus zu erhalten.

Wenn Sie ein Spiel mit Multitasking erstellen möchten: Vergessen Sie nicht, mit der ProgramPriority() Funktion eine hohe Priorität einzustellen, um mehr Rechenzeit zu erhalten.

## 1.10 showbitmap

### SYNTAX

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

### STATEMENT

Zeigt die angegebene Bitmap (#BitMap) auf dem gewünschten Bildschirm (ScreenID) an der Position x, y an. Diese Funktion ist 100% OS-freundlich und erlaubt schnelles Double-Buffering.

Wenn Sie ein Spiel mit Multitasking erstellen möchten: Vergessen Sie nicht, mit der ProgramPriority() Funktion eine hohe Priorität einzustellen, um mehr Rechenzeit zu erhalten.

---