

**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 V1.00 . . . . .	1
1.2	allocatebitmap . . . . .	1
1.3	allocatelinearbitmap . . . . .	2
1.4	freebitmap . . . . .	3
1.5	initbitmap . . . . .	3
1.6	bitmapid . . . . .	3
1.7	bitmapprastport . . . . .	3
1.8	usebitmap . . . . .	4
1.9	showbackbitmap . . . . .	4
1.10	showbitmap . . . . .	4

## Chapter 1

# BitMap

### 1.1 BitMap V1.00

PureBasic - BitMap library V1.00

Un 'BitMap' est une zone de mémoire utilisée pour stocker et/ou pour afficher des informations graphiques. Sur Amiga, le système natif d'affichage (OCS/ECS/AGA) utilise ce mode d'affichage dit par plans ('Planar') car chaque pixel est codé sur plusieurs plans (Si l'écran a plus de 2 couleurs). Dans un 'BitMap', on peut trouver 1 ou plusieurs 'BitPlanes'. Chaque 'BitPlane' a une surface équivalente à la taille de l'écran (exemple: 320\*200) et peut contenir pour chaque point, soit 0, soit 1 (un bit est un booléen, ne pouvant prendre que les valeurs 0 ou 1). Si on superpose plusieurs 'BitPlanes' les uns sur les autres, on peut avoir plus de possibilités pour chaque point: Pour 4 'BitPlanes' on aura  $2^4$  possibilité soit 16 valeurs différentes par points (dans notre cas, 16 couleurs différentes).

Commandes disponibles:

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

Exemple:

```
Double buffering
```

### 1.2 allocatebitmap

---

#### Syntaxe

```
BitMapID.1 = AllocateBitMap(#BitMap, Width, Height, Depth)
```

#### Résumé

Alloue une zone mémoire définie par la largeur, la hauteur et la profondeur du BitMap. Cette fonction est compatible avec les cartes graphiques (CyberGFX et Picasso 96).  
Si la valeur retournée par cette fonction est NULLE, alors le BitMap n'a pas pu être créé et par conséquent aucune opération ne doit être effectuée par la suite sur ce BitMap.

#BitMap: Identifiant numérique qui servira à reconnaître ce bitmap ultérieurement

Width : Largeur du BitMap en pixels

Height: Hauteur du BitMap en pixels

Depth : Profondeur du BitMap (nombre de BitPlanes). Cela définit le nombre de couleur que pourra supporter ce BitMap.  
Le calcul est le suivant: Nombre de couleurs =  $2^{\text{Depth}}$   
Donc un BitMap de Depth=5 pourra gérer 32 couleurs.

## 1.3 allocatelinearbitmap

#### Syntaxe

```
BitMapID.1 = AllocateLinearBitMap(#BitMap, Width, Height, Depth)
```

#### Résumé

Crée un nouveau bitmap en accord avec les paramètres donnés. Ce bitmap est un peu spécial, car il est composé que d'un seul bloc de mémoire. Il est alloué obligatoirement en mémoire CHIP et ne devrait servir que pour les conversions Chunky/Planar. C'est d'ailleurs dans ce seul but que cette fonction a été créée. Ce bitmap n'est pas compatible avec les cartes graphiques, donc si vous ne faites pas de chunky/planar utilisez la fonction `AllocateBitMap()`.

Si le résultat est NULL, il n'y a plus assez de mémoire pour allouer le bitmap.

#BitMap: Identifiant numérique qui servira à reconnaître ce bitmap ultérieurement

Width : Largeur du BitMap en pixels

Height: Hauteur du BitMap en pixels

Depth : Profondeur du BitMap (nombre de BitPlanes). Cela définit le nombre de couleur que pourra supporter ce BitMap.  
Le calcul est le suivant: Nombre de couleurs =  $2^{\text{Depth}}$   
Donc un BitMap de Depth=5 pourra gérer 32 couleurs.

---

## 1.4 freebitmap

Syntaxe

```
FreeBitMap(#BitMap)
```

Résumé

Libère l'espace memoire occupé par le BitMap. Aucune référence sur ce bitmap ne peut être faite ultérieurement.

#BitMap: Identifiant numérique représentant le BitMap précédemment créé par la fonction `AllocateBitMap()`

## 1.5 initbitmap

Syntaxe

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

Résumé

Initialise l'environnement nécessaire pour la gestion du nombre de BitMap passé en paramètre. Si la valeur retournée est NULLE, alors l'environnement ne peut pas être créé et vous ne devez en aucun cas utiliser une des fonctions de la bibliothèque 'BitMap'.

#NumBitMapMax: Nombre maximum de BitMap à gérer.

## 1.6 bitmapid

Syntaxe

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

Résumé

Retourne l'adresse mémoire du BitMap courant, qui sera parfois nécessaire pour les autre fonctions du PureBasic.

Note pour les programmeurs expérimentés: ce pointeur permet d'accéder aux informations sur le BitMap grâce à la structure 'BitMap' qui est definie dans les RKM du Rom3.0+.

Exemple: `*MyBitMap.BitMap = BitMapID()`

## 1.7 bitmaprastport

Syntaxe

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

Résumé

Retourne l'adresse du 'RastPort' du BitMap courant. Cette adresse est nécessaire pour certaines autre fonctions du PureBasic comme par exemple les fonctions de la bibliothèque '2D Drawing'. Ne vous inquiétez pas si

vous ne savez pas ce qu'est réellement un RastPort, ca n'a que peu d'importance. Il faut juste retenir que c'est un identifiant permettant de savoir où les graphiques vont être dessinés. Utilisez cette fonctions dès que vous avez besoin d'un 'RastPort'.

## 1.8 usebitmap

Syntaxe

```
UseBitMap(#BitMap)
```

Résumé

Change le BitMap courant par le nouveau BitMap passé en paramètre.

#BitMap: Identifiant numérique représentant le nouveau BitMap à définir comme courant.

## 1.9 showbackbitmap

Syntaxe

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

Résumé

Affiche le contenu du BitMap (#BitMap) dans le fond d'un écran de type 'DualPlayfield'. Si cet écran n'est pas 'DualPlayfield' alors utilisez la commande ShowBitMap(). Pour plus d'information sur le mode 'DualPlayfield' reportez vous à la documentation sur les Ecrans .

#BitMap: Identifiant du BitMap à afficher.

ScreenID: Identifiant de l'écran qui doit afficher le BitMap. Cette valeur peut être obtenue rapidement et simplement par la fonction ScreenID().

x,y : Position à laquelle doit être affiché le BitMap par rapport à l'écran (en pixels). La valeur 0,0 se situe en haut à gauche de l'écran.

## 1.10 showbitmap

Syntaxe

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

Résumé

Affiche le contenu du BitMap (#BitMap) sur l'écran (ScreenID) à la position x, y. Cette fonction est respecte le système Amiga et permet de faire du 'double buffering' (ou même du 'triple buffering' si nécessaire) très rapide.

#BitMap: Identifiant du BitMap à afficher.

---

ScreenID: Identifiant de l'écran qui doit afficher le BitMap. Cette valeur peut être obtenue rapidement et simplement par la fonction ScreenID().

x,y : Position à laquelle doit être affiché le BitMap par rapport à l'écran (en pixels). La valeur 0,0 se situe en haut à gauche de l'écran.