Bitmap

Paul Manias

COLLABORATORS			
	TITLE : Bitmap		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Paul Manias	May 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Bitmap

1.1 Object: Bitmap

```
OBJECT DOCUMENTATION
Name:      BITMAP
Version:   0.9 Beta.
Date:      December 1997
Author:    Paul Manias
Copyright: DreamWorld Productions, 1996-1997.  All rights reserved.

CHANGES VERSION 0.9B
Added:   Bitmap->AmtColours

Renamed: Bitmap->ScrType     to Bitmap->Type
         Bitmap->Owner       to Bitmap->Parent
         Bitmap->PlaneModulo to Bitmap->PlaneMod
         Bitmap->LineModulo  to Bitmap->LineMod

Edited:  Bitmap->Data
         Bitmap->ByteWidth
         Bitmap->Planes
         Bitmap->Type
         Bitmap->AmtColours
```

1.2 Object: Bitmap

```
OBJECT
Name:    Bitmap
Module:  Blitter
Version: 1
Type:    Descriptor

DESCRIPTION
The Bitmap provides a way of describing an area of memory that an
application can draw to, and/or display if the data is held in video
memory.  Bitmaps are most often used in the handling of Screens and Bobs,
and are required by most drawing functions such as DrawPixel().
```

```
ACTIONS
Actions supported by the Bitmap object are:

    Free()
    Get()
  * Init()

FUNCTIONS
Some of the functions supporting the Bitmap object are:

    ClearBitmap()
    DrawPixel()
    DrawLine()

STRUCTURE
The Bitmap structure consists of the following fields:

    AmtColours Amount of colours available.
    ByteWidth  Width in bytes.
    Data       Pointer to Bitmap data area.
    Height     Height in pixels.
    LineMod    Line differential.
    MemType    Memory type to use in allocations.
    Parent     Bitmap owner.
    PlaneMod   Plane differential.
    Planes     Amount of bitplanes.
    Restore    Restore list for this Bitmap, if any.
    Size       Total size of the Bitmap in bytes.
    Type       Display type.
    Width      Width in pixels.
```

1.3 Bitmap: AmtColours

```
FIELD
Name:        AmtColours
Type:        LONG
Inheritance: ScrType, Planes, User default.
Status:      Read/Write

DESCRIPTION
This  field  specifies the total amount of colours available in the Bitmap.
If  you  set  this value to NULL then Init() will fill it in for you, via a
check to the Planes and Type fields.
```

1.4 Bitmap: ByteWidth

```
FIELD
Name:        ByteWidth
Type:        WORD
Inheritance: Bitmap->Width
Status:      Read Only.
```

```
DESCRIPTION
The  ByteWidth  of  a Bitmap is calculated directly from the Bitmap's Width
and  Type settings.  Under no circumstances should you attempt to calculate
this value yourself, as it is heavily dependent on the Type.

Here are some of the calculations used, arranged by Bitmap Type:

Planar      = Width/8
ILBM        = Width/8
Chunky8     = Width
Chunky16    = Width * 2
True Colour = Width * 4

SEE ALSO
Field: Width
```

1.5 Bitmap: Data

```
FIELD
Name:        Data
Type:        APTR
Inheritance: Allocated on initialisation.
On Change:   Dynamic -- Parent objects do not usually like changes though,
             and you cannot change this field if it was set by Init().
Status:      Read/Write

DESCRIPTION
This  field  points  directly  to  the  start  of  the  Bitmap's data area.
Allocating  your  own  Bitmap  memory  is  acceptable  if you use MEM_DATA,
MEM_VIDEO  or  MEM_BLIT  types.  However  it  is  often best for Init() to
allocate  the  correct  amount  of  memory for you by leaving this field as
NULL.

The amount of memory allocated is dependant on the Size field.

SEE ALSO
Field: MemType
       Size
```

1.6 Bitmap: Height

```
FIELD
Name:        Height
Type:        WORD
Inheritance: Internal default.
On Change:   Cannot change after initialisation.
Status:      Read/IWrite

DESCRIPTION
The  pixel  height  of  of  the Bitmap is specified here.  It has a minimum
value of 1 and a maximum value of 32,767.
```

SEE ALSO
Field: Width

1.7 Bitmap: LineMod

FIELD
Name: LineMod
Inheritance: Width, Height and Planes settings.
On Change: Cannot change after initialisation.
Status: Read Only.

DESCRIPTION

SEE ALSO
Field: PlaneMod

1.8 Bitmap: MemType

FIELD
Name: MemType
Default: MEM_VIDEO
On Change: Cannot change after initialisation.
Status: Read/IWrite

DESCRIPTION
This field decides what type of memory will be allocated if Init() finds a Data pointer of NULL. This field uses the standard memory flags of MEM_DATA, MEM_VIDEO and MEM_BLIT. You may also specify the MEM_PRIVATE flag if necessary.

1.9 Bitmap: Parent

FIELD
Name: Parent
Inheritance: Init()
On Change: Cannot change after initialisation.
Status: Read/IWrite

DESCRIPTION
This abstract field points to the owner of the Bitmap. The pointer may be to any object but must be relevant for the situation. For example if the Bitmap is owned by a screen, then a pointer to that screen is placed here. If not, then a pointer to the Task that initialised the Bitmap will probably be found here.

1.10 Bitmap: PlaneMod

```
FIELD
Name:        PlaneMod
Inheritance: Width, Height and Planes settings.
On Change:   Cannot change after initialisation.
Status:      Read Only.

DESCRIPTION



SEE ALSO
Field: LineMod
```

1.11 Bitmap: Planes

```
FIELD
Name:        Planes, Depth, BitsPerPixel
Inheritance: Internal default.
On Change:   Cannot change after initialisation.
Status:      Read/IWrite

DESCRIPTION
This  field  specifies the amount of bitplanes (or bits per pixel depending
on  the  screen type) for the Bitmap.  The amount of colours you can use is
completely  dependent on this value.  For interleaved or planar bitmaps you
can  calculate  the amount of colours you get with the formula 2^n, where n
is the amount of planes.

For  chunky-based bitmaps, this field will be set to the amount of bits per
pixel - 8, 16 or 32 depending on the Type.

SEE ALSO
Field: Type
```

1.12 Bitmap: Size

```
FIELD
Name:        Size
Inheritance: Width, Height, and Depth calculation.
On Change:   Cannot change after initialisation.
Status:      Read Only.

DESCRIPTION
You  can read this field to find out how big the Bitmap is in bytes.  Never
write to this field, Init() will calculate it for you.

The calculation uses the formula:

     Size = Width * Height * Depth
```

1.13 Bitmap: Type

```
FIELD
Name:        Type
Type:        WORD
Inheritance: User Default.
On Change:   Cannot change after initialisation.
Status:      Read/IWrite

DESCRIPTION
The display data type - either PLANAR, INTERLEAVED, CHUNKY8, CHUNKY16 or
TRUECOLOUR. Descriptions of these display types are out of the scope of
this document, so if you require further information perhaps you should try
the RKM's. Note that for planar bitmaps, the bitplanes are stored
sequentially, one after the other. There is no scattering of planar
bitplane memory.

If you set this field to NULL then Init() will initialise it to the
preferred user screen type. This is exceptionally useful, as some types
are faster than others for certain effects. Remember that if you specify a
Type that is unsupported in hardware, then it will have to be emulated
(slowing things down considerably). Type independence is strongly
encouraged because of these reasons.
```

1.14 Bitmap: Width

```
FIELD
Name:        Width
Type:        WORD
Inheritance: Internal default.
On Change:   Cannot change after initialisation.
Status:      Read/IWrite

DESCRIPTION
The pixel Width of the bitmap is specified here. The width must be
divisible by 16, and cannot be less than 16 pixels across.

SEE ALSO
ByteWidth
Height
```

1.15 Object: Bitmap

```
ACTION
Name:   Init()
Object: Bitmap
Short:  Initialises a Bitmap object.

DESCRIPTION
This action will initialise a Bitmap object so that it is ready for use.
If the Bitmap->Data field has not been specified, a memory block will be
allocated and placed in this field. The type of memory allocated is
```

dependent on the Bitmap->MemType field. If you have not specified a memory type, you will get a default of MEM_DATA.

For this function to work properly you must have defined the Width and Height fields of the Bitmap before initialisation.

SEE ALSO
Kernel: Free()