

Bob_MBob

Paul Manias

COLLABORATORS

	<i>TITLE :</i> Bob_MBob		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Paul Manias	July 26, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Bob_MBob	1
1.1	Objects: Bobs, MBobs	1
1.2	Objects: Bob, MBob, BobEntry	1
1.3	Structure: BobEntry	3
1.4	MBob: AmtEntries	3
1.5	Bob/MBob: AmtFrames	3
1.6	Bob/MBob: Attrib	4
1.7	Bob/MBob: Bitmap	5
1.8	Bob/MBob: Buffers	5
1.9	Bob/MBob: ByteWidth	5
1.10	Bob/MBob: ClipLX, ClipTY, ClipRX, ClipBY	6
1.11	Bob/MBob: DirectGfx, DirectMasks	6
1.12	MBob: EntryList	7
1.13	MBob: EntrySize	7
1.14	Bob/MBob: Fplane	8
1.15	Bob: Frame	8
1.16	Bob/MBob: GfxCoords, MaskCoords	9
1.17	Bob/MBob: GfxData, MaskData	9
1.18	Bob/MBob: Height	10
1.19	Bob/MBob: Picture	10
1.20	Bob/MBob: Planes	11
1.21	Bob/MBob: PlaneSize	11
1.22	Bob/MBob: PropHeight	11
1.23	Bob/MBob: PropWidth	12
1.24	Bob/MBob: Screen	12
1.25	Bob/MBob: SrcMaskWidth	13
1.26	Bob/MBob: SrcWidth	13
1.27	Bob/MBob: Width	13
1.28	Bob: XCoord, YCoord	14
1.29	Bob/MBob: Init()	14

Chapter 1

Bob_MBob

1.1 Objects: Bobs, MBobs

OBJECTS DOCUMENTATION

Name: BOB & MBOB
Version: 0.9 Beta
Date: December 1997
Author: Paul Manias
Copyright: DreamWorld Productions, 1996-1997. All rights reserved.

CHANGES V0.9B

Removed: Bob->PictureTags
Edited: Bob->Picture
Bob->GfxCoords/MaskCoords
Bob->Clipxx
Bob->Planes
Bob->Picture

1.2 Objects: Bob, MBob, BobEntry

OBJECTS

Name: Bob	Name: MBob
Version: 1	Version: 1
Module: Blitter	Module: Blitter
Type: Complex	Type: Complex

DESCRIPTION

The Bob and MBob objects were created to provide simple but very powerful drawing possibilities in the Games Master System. For those that are not aware, "Bob" stands for Blitter Object and "MBob" stands for Multiple Blitter Object.

The drawing operations imposed on both structures have been simplified to allow more advanced features such as CPU assisted blitting. Also, there are many graphics boards with blitters that only perform simple copy operations. For this reason there are no special minterms available for these objects. Drawing operations include support for copying, drawing with masks, clearing and background restores.

In terms of structure, MBob's share the same properties as Bob's, except the design allows you to draw many graphics at once. Because an MBob has a virtually identical design to the Bob structure, the functions treat them as one and the same (eg DrawBob() also draws MBob's).

ACTIONS

The Bob and MBob objects support the following actions:

```

    Clear()
    Draw()
    Get()
*   Init()
    Free()

```

FUNCTIONS

The Blitter module supports the following functions for Bobs and MBobs:

```

ClearBob()
CreateMasks()
DrawBob()
DrawBobList()
SetBobFrames()
SetBobDrawMode()
SetBobDimensions()

```

STRUCTURE

To fully understand what Bob and MBob's are capable of, it is highly recommended that you read this next section thoroughly.

AmtEntries	Amount of entries in the image list (MBob).
AmtFrames	Total amount of frames in GfxCoords.
Attrib	Flags like CLIP and MASK.
Bitmap	Destination Bitmap.
Buffers	Amount of buffers in dest screen.
ByteWidth	Graphic width in bytes.
ClipLX	Left X border.
ClipTY	Top Y border.
ClipRX	Right X border.
ClipBY	Bottom Y border.
DirectGfx	Pointer to direct frame list.
DirectMasks	Pointer to direct mask list.
EntryList	Pointer to entry list (MBob).
EntrySize	Size of each entry (MBob).
FPlane	First plane to blit to (planar only).
Frame	Current frame number (Bob).
GfxCoords	Pointer to frame list for graphics.
GfxData	Pointer to Bob graphics.
Height	Height in pixels.
MaskCoords	Pointer to frame list for masks.
MaskData	Pointer to Bob masks.
+ Picture	Pointer to picture object source.
Planes	Amount of planes.
PlaneSize	Size of plane source in bytes (planar only).
PropWidth	Expected width of source picture.
PropHeight	Expected height of source picture.
Screen	Destination Screen.

SrcWidth	Source page width in bytes.
SrcMaskWidth	Mask page width in bytes.
Width	Pixel width.
XCoord	X Coordinate of the object (Bob).
YCoord	Y Coordinate of the object (Bob).

1.3 Structure: BobEntry

Name: BobEntry
Version: 1
Type: Child Descriptor
Parent: MBob

DESCRIPTION

1.4 MBob: AmtEntries

FIELD
Name: AmtEntries
Type: WORD
Inheritance: Defaults to 1 if unspecified.
On Change: Dynamic
Status: Read/Write

DESCRIPTION

Specifies the amount of images to be blitted when you call a function that draws MBobs. The value can be set dynamically and does not require initialisation by Init().

This field is only available in MBob structures.

SEE ALSO

Field: EntryList

1.5 Bob/MBob: AmtFrames

FIELD
Name: AmtFrames
Type: WORD
Inheritance: GfxCoords, GfxDirect.
On Change: Cannot change after initialisation.
Status: Read Only.

DESCRIPTION

This field specifies the amount of frames present in the Bob. Writing to this field is disallowed. Init() will calculate it by adding up all the

frames present in the GfxCoords or GfxDirect lists (minimum value of 1).

SEE ALSO

Field: GfxCoords
GfxDirect

1.6 Bob/MBob: Attrib

FIELD

Name: Attrib
Type: LONG
On Change: SetBobDrawMode()
Status: Read Only.

DESCRIPTION

Specifies the attributes used in the blitting of this Bob, as well as some special flags that define the way the Bob will be initialised. The current flags are:

CLIP

Allow clipping for this Bob, to the regions given in ClipTX, ClipTY, ClipRX and ClipBY. Note that clipping does slow down the drawing procedure so omit this flag if possible.

MASK

Allow masking for this Bob.

GENMASKS

Creates a mask for every frame of your Bob.

FILLMASK

Fills any "empty holes" in the mask, so that no background graphics can show through the middle areas of the bob. This can be useful in certain situations, eg the Reko module uses it so that only one mask is needed for all the Cards.

RESTORE

Specified if this bob is to be added to a Restore object each time it is drawn. This allows automatic background restoring on the Bob when the buffer returns from the display. For more information see the Restore object.

CLEAR

Specified if this Bob is to be added to a Restore object each time it is drawn. This allows automatic background clearing on the Bob when the buffer returns from the display. For more information see the Restore object.

CLRMASK

A mask will be used if you clear this bob.

CLRNOMASK

No mask will be used if you clear this Bob (default).

1.7 Bob/MBob: Bitmap

FIELD

Name: Bitmap
Type: struct Bitmap *
Inheritance: Init()
On Change: Cannot change after initialisation.
Status: Read Only.

DESCRIPTION

This field identifies the Bitmap that the Bob is being blitted to. It is a compulsory setting which is inherited from the container argument for Init().

SEE ALSO

Object: Bitmap

1.8 Bob/MBob: Buffers

FIELD

Name: Buffers
Type: WORD
Inheritance: Screen
Default: 1
On Change: Cannot change after initialisation.
Status: Read/Write

DESCRIPTION

This field specifies the amount of buffers in use by the Bob's Screen. If your Bob was not initialised to a Screen, you will have to set this field yourself if you want buffer handling for your Bob.

The minimum value is 1 and the maximum value is 3.

NOTE

It is very important that this field is set correctly when using a Restore object with your Bob.

1.9 Bob/MBob: ByteWidth

FIELD

Name: ByteWidth
Type: WORD
Inheritance: Width
On Change: SetBobDimensions()
Structures: Bob, MBob
Status: Read only.

DESCRIPTION

The width of the bob in bytes. This field cannot allow uneven byte values such as 1, 3, 5... only values of 2, 4, 6... The maximum value currently supported by this field is 128 bytes.

Do not write to this field, allow Init() to calculate it from your Width setting.

SEE ALSO

Field: Width

1.10 Bob/MBob: ClipLX, ClipTY, ClipRX, ClipBY

FIELDS

Name: ClipLX, ClipTY, ClipRX, ClipBY
Type: WORD
Inheritance: Screen->Width, Screen->Height.
On Change: Cannot change after initialisation.
Status: Read/Write

DESCRIPTION

These fields set the clipping area for your bob. The values that you specify must be within the borders of your Bitmap's area, which means no negative values or values that exceed the Height and Width of the destination bitmap.

ClipLX and ClipRX values must be specified in increments of 16 pixels for compatibility with the planar style screen modes.

Dynamic changes to these fields are not permitted.

1.11 Bob/MBob: DirectGfx, DirectMasks

FIELDS

Name: DirectGfx, DirectMasks
Type: LONG *
Inheritance: Generated from GfxCoords, MaskCoords, GfxData and MaskData.
On Change: Dynamic
Status: Read/Write

DESCRIPTION

These fields point to lists of addresses that give pointers to graphic or mask data. Normally you will not need to create your own lists, as Init() will generate them through the *Coords, *Data and Picture fields. A direct list looks like this:

```
dc.l <Address>
dc.l ...
dc.l -1
```

EXAMPLE:

GfxFrames:

```
dc.l GFX_Sparkie+00
dc.l GFX_Sparkie+20
dc.l GFX_Sparkie+40
```

```
dc.l    GFX_Sparkie+60
dc.l    -1
```

The very first address pointer is considered to be frame 0, the next is frame 1, and so on.

NOTE

If you specify the GENMASK (Generate Mask) flag as an attribute, make sure that you do not supply a list in DirectMasks field (it will be over written).

SEE ALSO

```
Fields: GfxCoords
        GfxData
```

1.12 MBob: EntryList

FIELD

```
Name:      EntryList
Type:      APTR
On Change: Dynamic
Status:    Read/Write/Compulsory
Object:    MBob
```

DESCRIPTION

If you are using an MBob, you will point to an EntryList of images to be blitted here. There is no termination necessary for your list, but the MBob->AmtEntries field must be specified correctly.

The Entry structure looks like this:

```
struct MBEntry {
    WORD XCoord;    /* X Coordinate */
    WORD YCoord;    /* Y Coordinate */
    WORD Frame;     /* Frame no. for this Entry */
};
```

If you specify SKIPIMAGE in an X Coordinate then that particular image will not be drawn to screen.

This field is only available for MBob structures.

SEE ALSO

```
Field: EntrySize
```

1.13 MBob: EntrySize

FIELD

```
Name:      EntrySize
Type:      WORD
Default:   BE_SIZEOF
On Change: Dynamic
```

Structure: MBob
Status: Read/Write

DESCRIPTION

Here you must specify the byte size that is taken up by each entry in your MBob->EntryList. This allows you to create mutated entry structures, which you can use to store extra data for each image in your EntryList. For more information see mutant structure types.

As a default this field will be set to MBE_SIZEOF (the minimum size).

SEE ALSO

Field: EntryList

1.14 Bob/MBob: Fplane

FIELD

Name: FPlane
Type: WORD
On Change: Dynamic
Status: Read/Write

DESCRIPTION

Specifies the first plane that is going to be blitted to. This field is relevant for planar screen types only.

SEE ALSO

Field: Planes

1.15 Bob: Frame

FIELD

Name: Frame
On Change: Dynamic
Status: Read/Write
Object: Bob

DESCRIPTION

Specifies the frame number to use when drawing your bob. This field is ignored in the initialisation procedure, but is used for Draw() and other Bob drawing functions. The values for this field range from 0 to the maximum frame specified in the AmtFrames field. The main use of the Frame field is to allow you to give animation properties to a Bob.

Note that this field is only available in the Bob object. For MBob's, the frame number is set in the EntryList.

SEE ALSO

Field: GfxCoords

1.16 Bob/MBob: GfxCoords, MaskCoords

FIELDS

Name: GfxCoords, MaskCoords
Type: struct FrameList *
On Change: SetBobFrames()
Status: Read/Write

DESCRIPTION

You need to point these fields to FrameList arrays if you want to generate the graphics and mask pointers for your Bob. The FrameList structure is very simple and looks like this:

```
struct FrameList {  
    WORD XCoord;  
    WORD YCoord;  
};
```

These coordinates will be used in conjunction with the GfxData and MaskData fields in order to generate your Direct* pointer arrays. If you are going to supply your own DirectGfx and DirectMask lists (this is rare), you can ignore the FrameList based fields.

Note that the X values should be aligned to every 16th pixel, and that negative values are disallowed. The format for a FrameList looks like this:

```
dc.w <XCoord>,<YCoord>  
dc.w ...  
dc.w -1,-1
```

An example:

```
GfxFrames:  
dc.w 0,10  
dc.w 0,26  
dc.w -1,-1
```

If you make a change to your FrameList/s after calling Init(), you have to call SetBobFrames(). This will re-write the changes to DirectGfx and DirectMasks. Note that you must never attempt to extend or shrink the FrameList after calling Init(). Such actions can have adverse effects on other areas of a Bob's functionality.

SEE ALSO

Field: AmtFrames
DirectGfx
GfxData
Blitter: SetBobFrames()

1.17 Bob/MBob: GfxData, MaskData

FIELD

Names: GfxData, MaskData

Type: APTR
Inheritance: Picture->Data
On Change: SetBobFrames()
Status: Read/IWrite

DESCRIPTION

These fields contain pointers to the Bob's graphic and mask origins (raw data). When used in conjunction with the GfxCoords and MaskCoords arrays, Init() will be able to calculate the DirectGfx and DirectMask pointers for each frame. This saves a lot of effort as the DirectGfx and DirectMask fields can be difficult to calculate in most circumstances.

You do not need to supply these values if you are already supplying addresses in DirectGfx and DirectMasks, or if you are supplying a Picture source.

SEE ALSO

Field: GfxCoords

1.18 Bob/MBob: Height

FIELD

Name: Height
Type: WORD
Inheritance: Picture->Height
On Change: SetBobDimensions()
Status: Read/Write

DESCRIPTION

The height of the Bob in pixels. If you change this field, you must immediately make a call to SetBobDimensions() to register the change.

SEE ALSO

Field: Width

1.19 Bob/MBob: Picture

FIELD

Name: Picture
Type: struct Picture *
Inheritance: None
On Change: Cannot change after initialisation.
Status: Read/IWrite

DESCRIPTION

This field specifies the Picture object that the Bob originated from. If the Bob used no Picture source, then this field will remain as NULL.

Once a Bob is initialised from a Picture, the Bob will retain its link to it. Keep in mind that if you free a Picture that is linked to one or more Bobs, you must free those Bobs as well, or you will run into trouble very quickly.

1.20 Bob/MBob: Planes

FIELD

Name: Planes, Depth, BitsPerPixel
Type: WORD
Inheritance: Picture->Planes
On Change: Dynamic
Status: Read/Write

DESCRIPTION

Describes the amount of planes used by this bob. This field will be initialised to the amount of planes in the Picture if set at zero.

For chunky style Types such as True Colour, this field will reflect the amount of bits per pixel.

SEE ALSO

FPlane

1.21 Bob/MBob: PlaneSize

FIELD

Name: PlaneSize
Type: LONG
Inheritance: Picture
On Change: Cannot change after initialisation.
Status: Read Only.

DESCRIPTION

The plane size for this Bob, calculated by PageWidth*PageHeight. This field is only relevant for planar Bob's with more than 1 plane to be blitted.

1.22 Bob/MBob: PropHeight

FIELD

Name: PropHeight
Type: WORD
On Change: Cannot change after initialisation.
Status: Write Only.

DESCRIPTION

By setting this field you will enable proportional resizing for the Bob image. For this to be of use, you need to be initialising the Bob with a Picture object as the source, as Init() will be utilising the RESIZE image flag.

The theory is that you set the PropHeight field to the "expected" height of the source picture. If the PropHeight does not match the Picture->Height, then the Bob->Height will be recalculated to reflect this change in the source. This basically means that the Bob will expand or shrink due to its sensitivity of the source picture's dimensions.

The formula used is thus:

$$\text{Bob} \rightarrow \text{Height} = (\text{Picture} \rightarrow \text{Height}) / (\text{Bob} \rightarrow \text{PropHeight} / \text{Bob} \rightarrow \text{Height})$$

SEE ALSO

Field: PropWidth

1.23 Bob/MBob: PropWidth

FIELD

Name: PropWidth

Type: WORD

On Change: Cannot change after initialisation.

Status: Write Only.

DESCRIPTION

By setting this field you will enable proportional resizing for the Bob image. For this to be of use, you need to be initialising the Bob with a Picture object as the source, as Init() will be utilising the RESIZE image flag.

The theory is that you set the PropWidth field to the "expected" width of the source picture. If the PropWidth does not match the Picture->Width, then the Bob->Width will be recalculated to reflect this change in the source. This basically means that the Bob will expand or shrink due to sensitivity of the source picture's dimensions.

The formula used is thus:

$$\text{Bob} \rightarrow \text{Width} = (\text{Picture} \rightarrow \text{Width}) / (\text{Bob} \rightarrow \text{PropWidth} / \text{Bob} \rightarrow \text{Width})$$

SEE ALSO

PropHeight

1.24 Bob/MBob: Screen

FIELD

Name: Screen

Type: struct Screen *

Inheritance: Init()

On Change: Cannot change after initialisation.

Status: Read Only.

DESCRIPTION

This field contains a pointer to the Screen that owns the Bob. A Bob does not necessarily need to belong to a Screen (a Bitmap is required as minimum) so a NULL setting in this field is not out of the ordinary here.

SEE ALSO

Field: Bitmap

1.25 Bob/MBob: SrcMaskWidth

FIELD

Name: SrcMaskWidth
Type: WORD
Inheritance: Picture->Width or Bob->ByteWidth
On Change: Cannot change after initialisation.
Status: Read/Write

DESCRIPTION

This field specifies the byte width of the Bob's mask page area. This value is important for correctly pre-calculating the Bob's modulo for the Draw*() functions. If the mask was cut out of a Picture (eg as a brush), this value will generally be the same as the one in Bob->ByteWidth. If you are blitting the Bob directly from a Picture, it will be inherited from Picture->Width.

This value should never be set on initialisation unless absolutely necessary. Most developers should have no use for this field, as it is mostly needed for internal purposes.

SEE ALSO

Field: SrcMaskWidth

1.26 Bob/MBob: SrcWidth

FIELD

Name: SrcWidth
Type: WORD
Inheritance: Picture->Width or Bob->ByteWidth
On Change: Cannot change after initialisation.
Status: Read/Write

DESCRIPTION

This field specifies the byte width of the Bob's page area. This value is important for correctly pre-calculating the Bob's modulo for the Draw*() functions. If the Bob was cut out of a picture (eg as a brush), this value will generally be the same as the one in Bob->ByteWidth. If you are blitting the bob directly from a Picture, it will be inherited from Picture->Width.

This value should never be set on initialisation unless absolutely necessary. Most developers should have no use for this field, as it is mostly needed for internal purposes.

SEE ALSO

Field: SrcMaskWidth

1.27 Bob/MBob: Width

FIELD

Name: Width

Type: WORD
Inheritance: Picture->Width
On Change: SetBobDimensions()
Status: Read/Write

DESCRIPTION

The width of the Bob in pixels. There is no need to align this field to every 16th pixel. If you change this field, you must immediately make a call to SetBobDimensions() to register the change.

SEE ALSO

Field: Height

1.28 Bob: XCoord, YCoord

FIELDS

Name: XCoord, YCoord
Type: WORD
On Change: Dynamic
Status: Read/Write

DESCRIPTION

This field specifies the X and Y positions of the Bob in relation to the destination Bitmap.

You are allowed to specify coordinates that exceed the boundaries of the destination Bitmap, but only if you have specified the CLIP flag as an Attribute (in which case the Bob will be clipped appropriately).

If you omit the CLIP flag and then place the Bob at the screen borders, you will risk blitting the Bob to unknown video ram areas.

This field is only available in Bob objects.

1.29 Bob/MBob: Init()

ACTION

Name: Init()
Synopsis: Init({Bob}|{MBob}, {Screen}|{Bitmap})
Objects: Bob, MBob.
Short: Initialises a Bob or MBob so that it is ready for blitting.

DESCRIPTION

Initialises a Blitter object to a Bitmap container. If you initialise a Bob or MBob using tags, make sure that you specify the correct type of TAGS_BOB or TAGS_MBOB. If the type is incorrect, you will confuse the function and not get what you want.

SEE ALSO

Kernel: Free()
