The symbol  is used to warn of features not available in all versions of PC GEM.

The order of versions released is:

- GEM/1
- GEM/2
- GEM/3
- ViewMAX/1
- ViewMAX/2
- ViewMAX/3 beta ("Panther", never released)
- The AES build available from these pages. It includes all ViewMAX features; so "ViewMAX/2 and later" includes this AES.

Structures used by PC GEM

OBJECT

typedef struct object
{

WORD ob_next;
Number of the object's next sibling; if none, -1 (NIL).
WORD ob_head;
Object's first child; if none, -1.
WORD ob_tail;
Object's last child; if none, -1.
UWORD ob_type;
The object type. The high byte of this word is available to the programmer for whatever use they want; the low byte holds the type used by the AES.
UWORD ob_flags;
The object flags. These generally do not change through an object's lifetime.
UWORD ob_state;
The object's state bits. These may change.
LONG ob_spec;
This value is polymorphic; either a far pointer to information, or a 32-bit integer containing various bitfields.
UWORD ob_x;
Upper left corner of object - X
UWORD ob_y;
Upper left corner of object - Y
UWORD ob_width;
Width of object
UWORD ob_height;
Height of object

} OBJECT;

The first three members of **OBJECT** are used to form trees of objects. A tree is a linear array of objects, with their next/head/tail members used to construct the correct relationships.

Object types

The object type is a word, but the AES only uses the low 8 bits; the high 8 bits are available for use by the programmer. Object types are:

```
#define G_BOX 20        /* Solid rectangle */
#define G_TEXT 21       /* Formatted text */
#define G_BOXTEXT 22    /* Formatted text with border and background */
#define G_IMAGE 23      /* Bitmapped image */
#define G_USERDEF 24    /* Object drawn by program, not by AES
#define G_IBOX 25       /* Hollow rectangle */
#define G_BUTTON 26     /* Button */
#define G_BOXCHAR 27    /* Rectangle containing a single character */
#define G_STRING 28     /* Unformatted text */
#define G_FTEXT 29      /* Editable text field */
#define G_FBOXTEXT 30   /* Editable text field with border and background */
#define G_ICON 31       /* Icon (image + caption) */
#define G_TITLE 32      /* Menu titlebar entry */
#define G_CLRICN 33     /* Colour icon */
#define G_DTMFDB 34     /* For internal AES use only: desktop image */
```

 Object types 33 and 34 are only supported in the ViewMAX/3 beta and in FreeGEM.

Object flags

Object flags are bitmapped:

```
#define SELECTABLE 0x1  /* User can click to toggle the "selected" state */
#define DEFAULT 0x2     /* Button is default button */
#define EXIT 0x4        /* Selecting this object will leave form_do() loop */
#define EDITABLE 0x8    /* Object can have keyboard docus */
#define RBUTTON 0x10    /* Object is a radio button */
#define LASTOB 0x20     /* Object is the last object in the array */
#define TOUCHEXIT 0x40  /* Clicking this object will leave form_do() loop */
#define HIDETREE 0x80   /* This object and its children are not drawn */
#define INDIRECT 0x100  /* ob_spec is a far pointer to the specification,
                         * rather than the specification itself  */
#define ESCCANCEL 0x200 /* ESCape is a shortcut for this button */
#define BITBUTTON 0x400 /* Not implemented in any known AES */
#define SCROLLER 0x800  /* Button is an "up" or "down" scrollbar button */
#define FLAG3D 0x1000   /* Draw object in 3D  */
#define FL3DIND FLAG3D  /* for Atari compatibility */
#define USECOLORCAT 0x2000 /* Use ViewMAX predefined colour set */
#define FL3DBAK 0x4000  /* 3D background (sunken rather than raised) */
#define SUBMENU 0x8000  /* Not implemented in any known PC AES */
```

 Flags from ESCCANCEL to SUBMENU are only effective in ViewMAX/2 and later. These bits have other meanings in recent versions of Atari GEM.

If USECOLORCAT is set, the object's interior/background colour will be an index into the 16 ViewMAX colour categories. Buttons will always use no. 11.

If FLAG3D is set, USECOLORCAT is implied. ViewMAX buttons are always 3D.

 Later FreeGEM builds use FL3DBAK to implement a "sunken" 3D area. To check for this feature, use appl_init() and check that bit 3 of xbuf.abilities is set.

Object states

Object states are bitmapped:

```
#define SELECTED 0x1    /* If 3D, object appears "pressed"; else "inverted" */
#define CROSSED 0x2     /* Object has a white cross through it */
#define CHECKED 0x4     /* Object has an arrow in its top left-hand corner */
#define DISABLED 0x8    /* Object is greyed out */
#define OUTLINED 0x10   /* Object has an extra border around it */
#define SHADOWED 0x20   /* Object has a dropped shadow; depth = thickness of */
                        /*                                      object border */
#define WHITEBAK 0x40   /* Icon background is white, not transparent */
#define DRAW3D 0x80     /* Highlight icon by making it bold rather than */
                        /* inverse */
#define HIGHLIGHTED 0x100    /* Draw focus rectangle around object */
#define UNHIGHLIGHTED 0x200  /* Remove existing focus rectangle */
```

 HIGHLIGHTED and UNHIGHLIGHTED are only available in ViewMAX/2 and later versions.

 In GEM/5, CROSSED makes the object draw in 3D:

- If an object is both CROSSED and SELECTABLE, then it is drawn as a checkbox.
- If it is CROSSED, SELECTABLE and an RBUTTON, it is drawn as a radio button.
- If it is a button or a box and it is CROSSED, then it is drawn as a raised 3D shape, similar to Motif.
- If a button is CROSSED and DEFAULT, a "Return key" symbol appears on it (rather like NEXTSTEP).
- Boxes and text fields that are CROSSED and CHECKED appear sunken.

GEM/5 can be detected by calling vqt_name() for font 1. If nothing is returned, GEM/5 is running.

 Recent FreeGEM builds contain a system similar based on the GEM/5 one, but extended and backwards-compatible. The DRAW3D state is used instead of CROSSED:

- If an object is both DRAW3D and SELECTABLE, then it is drawn as a checkbox.
- If it is DRAW3D, SELECTABLE and an RBUTTON, it is drawn as a radio button.
- If a button is DRAW3D and DEFAULT, a "Return key" symbol will be on it.
- If an object with a 3D border has the WHITEBAK state, then the 3D border will not have a black edge.
- If a radio button or checkbox has the WHITEBAK state, then it will be drawn with a white background rather than in the colour used by 3D objects.

To check for these abilities, use appl_init() and check that bit 3 of xbuf.abilities is set.

Object specifications

The object specification varies depending on the type of the object being specified:

G_BOX, G_IBOX, G_BOXCHAR

The spec is a 32-bit word, laid out as follows:

- Bits 0-3: Interior colour
- Bits 4-6: Interior pattern (0=white, 7=black, others stipples)
- Bit 7: 0 if text should have transparent background, else 1.
- Bits 8-11: Text colour

- Bits 12-15: Border colour
- Bits 16-23: Border thickness - signed byte. Negative means border is within object's bounding rectangle; positive means it is outside.
- Bits 24-31: Character to display (G_BOXCHAR).

G_BOXTEXT, G_FBOXTEXT, G_TEXT, G_FTEXT
: The spec is a far pointer to a TEDINFO structure.

G_TITLE, G_STRING, G_BUTTON
: The spec is a far pointer to the text to display.

G_USERDEF
: The spec is a far pointer to a USERBLK structure.

G_ICON, G_CLRICN
: The spec is a far pointer to an ICONBLK structure.

G_IMAGE
: The spec is a far pointer to a BITBLK structure.

G_DTMFDB
: The spec is a far pointer to a MFDB structure.

ORECT

```
typedef struct orect
{
        struct orect *o_link;
        WORD    o_x;
        WORD    o_y;
        WORD    o_w;
        WORD    o_h;
} ORECT;
```

The ORECT is mainly used internally in the AES.

GRECT

```
typedef struct grect
{
        WORD    g_x;
        WORD    g_y;
        WORD    g_w;
        WORD    g_h;
} GRECT;
```

The GRECT is a general-purpose rectangle.

TEDINFO

```
typedef struct text_edinfo
{
BYTE far *te_ptext;
        /* pointer to text */
BYTE far *te_ptmplt;
        /* pointer to template */
BYTE far *te_pvalid;
```

/* pointer to to validation characters */
WORD te_font;
/* font (3=normal, 5=small) */
WORD te_junk1;
/* junk word */
WORD te_just;
/* justification: 0=left 1=centre 2=right */
WORD te_color;
/* colour information word: */

- Bits 0-3: Background colour
- Bits 4-6: Background pattern (0=white, 7=black, others stipples)
- Bit 7: 0 if text should have transparent background, else 1.
- Bits 8-11: Text colour
- Bits 12-15: Border colour

WORD te_junk2;
/* junk word */
WORD te_thickness;
/* border thickness */
WORD te_txtlen;
/* length of text string */
WORD te_tmplen;
/* length of template string */

} TEDINFO;

This specifies a (possibly editable) formatted string object.

"Professional GEM" states:

> One final note on editable text objects: GEM's editor uses the commercial at sign '@' as a "meta-character". If it is the first byte of the initialized text, then the field is displayed blank no matter what follows. This can be useful, but is sometimes confusing when a user in all innocence enters an @ and has his text disappear the next time the dialog is drawn!

ICONBLK

typedef struct icon_block
{
BYTE far * ib_pmask;
Address of mask bitmap (device-dependent form)
BYTE far * ib_pdata;
Address of image bitmap (device-dependent form)
BYTE far * ib_ptext;
Address of caption text
WORD ib_char;

- Bits 0-7: Character to superimpose on the icon
- Bits 8-11: Foreground colour of icon
- Bits 12-15: Background colour of icon

WORD ib_xchar;
X-coordinate of character relative to icon origin

WORD ib_ychar;
Y-coordinate of character relative to icon origin
WORD ib_xicon;
X-coordinate of image relative to icon origin
WORD ib_yicon;
Y-coordinate of image relative to icon origin
WORD ib_wicon;
Width of image
WORD ib_hicon;
Height of image
WORD ib_xtext;
X-coordinate of caption relative to icon origin
WORD ib_ytext;
Y-coordinate of caption relative to icon origin
WORD ib_wtext;
Width of caption area
WORD ib_htext;
Height of caption area

} ICONBLK;

In a colour icon, ib_pdata and ib_pmask point to MFDB objects. Otherwise, they point to the lines of the bitmap.

 The colour icon differs from Atari GEM, which stores colour plane information immediately after the ICONBLK structure.

BITBLK

typedef struct bit_block
{
BYTE far *bi_pdata;
/* Bitmap data, in device-dependent form */
WORD bi_wb;
/* width of data in bytes */
WORD bi_hl;
/* height in lines */
WORD bi_x;
/* X-coordinate of bitmap relative to object origin */
WORD bi_y;
/* Y-coordinate of bitmap relative to object origin */
WORD bi_color;
/* Foreground colour of bitmap */

} BITBLK;

USERBLK

typedef struct user_blk

{

VOID far *ub_code;
Drawing code for self-drawing object.
LONG ub_parm;
Application-defined parameter for the object.

} USERBLK;

The drawing code will be entered with AX:BX = address of parameter block. It should return in AX any "state" bits that it wants the built-in code to apply to the object after it is drawn; normally these will be 0.

The APPLBLK structure {ab_code, ab_parm} is sometimes used for the same purpose.

PARMBLK

typedef struct parm_blk
{

OBJECT far *pb_tree;
Tree containing the object to draw
WORD pb_obj;
Index of object within tree
WORD pb_prevstate;
Previous object **ob_state** word
WORD pb_currstate;
Current object **ob_state** word
WORD pb_x, pb_y, pb_w, pb_h;
Object bounding rectangle
WORD pb_xc, pb_yc, pb_wc, pb_hc;
Clipping rectangle
LONG pb_parm;
ub_parm from this object's USERBLK.

} PARMBLK;

CLRCAT

typedef struct clrcat
{

WORD cc_foreground;
Foreground colour
WORD cc_background;
Background colour
WORD cc_style;
Fill style
WORD cc_pattern;
Fill pattern

} CLRCAT;

The CLRCAT is used internally by ViewMAX/2 and later to store the colour categories. It is exposed by the X_BUF_V2 structure below.

X_BUF_V2

typedef struct x_buf_v2
{

WORD buf_len;
: Length of the structure, including this word. Future versions of this structure (X_BUF_V3 etc.) may be bigger.

WORD arch;
: 16 for 16-bit AES, 32 for hypothetical 32-bit AES.

CLRCAT far *cc;
: Address of an array of 16 CLRCAT structures. This is so that they can be read by a program; in ViewMAX, the colours could be set but not reread.

OBJECT far *w_active;
: Address of an object tree (19 elements) used to draw window elements. Included so a program can change symbols on window buttons.

BYTE far *info;
: Address of a 0-terminated ASCII string (at most 40 characters, no newlines) describing the AES

LONG abilities;
: A bitmapped field describing what optional functions this AES provides:
  - Bit 0: An Atari-style appl_getinfo() call is present.
  - Bit 1: prop_get(), prop_put() and prop_del() are present.
  - Bit 2: wind_get() and wind_set() can change window options.
  - Bit 3: Extended 3D support.
  - Bit 4: xshl_getshell() and xshl_setshell() are present.

  appl_getinfo() and prop_*() calls are compile-time options in my Pacific C AES.

} X_BUF_V2;

An initialised X_BUF_V2 is one in which all members are 0 except **buf_len**. This initialised buffer is then passed to appl_init(). On return, if **arch** is 0 then the structure was not filled in by the AES; otherwise it was. The **buf_len** field may be reduced, if the AES was expecting an earlier version of the structure (ie, X_BUF_V1); this should not be a problem because the structures are forward and backward compatible.

This text was originally created by John Elliott, and was located on his website www.seasip.info.
This version of the document was packaged by Shane M. Coughlan for the OpenGEM SDK.