AmigaMail

	COLLABORATORS		

	TITLE : AmigaMail		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 11, 2022	

REVISION HISTORY			

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmigaMail

1.1   VII-17: ECS Display Modes and ILBM CAMG

By Carolyn Scheppner

Under versions of the Amiga operating system and hardware prior to release
2.0, the display modes available to the system (such as HIRES, LACE, HAM,
HALFBRITE, DUALPF) could all be described in the 16-bit ViewPort.Modes
field. When saving an Amiga display to an ILBM file, the ViewPort mode
field, also referred to as a view mode, was preserved along with the image
to make sure it could be later displayed in the proper display mode. The
ILBM CAMG chunk stores a display's view mode.

When using Amiga operating systems prior to release 2.0, storing the view
mode in an ILBM involves extracting the 16-bit view mode, masking out the
GENLOCK_AUDIO, GENLOCK_VIDEO, VP_HIDE, and SPRITES flags, and storing the
result as a 32-bit value in the CAMG chunk. The upper word of the value
should be zero. Reading the view mode requires masking out the
undesirable bits mentioned above from the CAMG's 32-bit value, and then
use the low word of the result as a 16-bit value for ViewPort.Modes or
NewScreen.ViewModes when opening a display.

The 2.0 operating system and the ECS chips support a large number of
display modes--too many for the old 16-bit view mode to represent. Under
2.0, display modes are now specified by a 32-bit ModeID which is not
stored in the ViewPort, but is instead held in private graphics lists.
The graphics.library provides functions for accessing this extended
display information. The GetVPModeID() function returns a ViewPort's
32-bit ModeID:

    ULONG modeID = GetVPModeID(struct ViewPort *vp);

When saving a display under release 2.0, an ILBM writer must preserve the
entire 32-bit modeID in the CAMG chunk. The advantage of this is ILBM
readers that don't know about the release 2.0 32-bit view mode can still
display ILBMs saved under 2.0. Although ModeIDs are numeric values rather
than bit masks, the 2.0 ModeIDs were designed to contain the pre-2.0 bits
in their low word for the matching (or closest match) old ViewPort mode.
For example, the 2.0 ModeID for a Hires-Interlace display has the HIRES
and LACE bits set in its low word. The 2.0 ModeID for a Productivity

Interlaced display also has the HIRES and LACE bits set in its low word
because, of the display modes available to a non-ECS or pre-2.0 system, a
Hires-Interlace display is most like the Productivity Interlaced display.

By storing the entire 32-bit ModeID in the CAMG chunk, 2.0 compatible ILBM
readers can display the image in the proper display mode if that display
mode is available to the current system. Old ILBM readers will not be
able to use the new display modes, but they will truncate the new 32-bit
ModeID into a 16-bit view mode which will normally provide a reasonable
view mode for the image.

There are several CAMG values saved by old ILBM writers which are invalid
modeID values. An ILBM writer will produce ILBMs with invalid modeID
values if it:

    *   fails to mask out SPRITES|VP_HIDE|GENLOCK_AUDIO|GENLOCK_VIDEO.

    *   saves a 2.0 extended view mode as 16 bits rather than saving the
        32-bit modeID returned by GetVPModeID().

    *   saves garbage in the upper word of the
        CAMG value.

All valid modeIDs either have an upper word of 0 and do not have the
<graphics/view.h> EXTENDED_MODE bit set in the low word, or have a non-0
upper word and do have the EXTENDED_MODE bit set in the lower word. CAMG
values which are invalid modeIDs must be screened out and fixed before
using them.

```
/* Knock bad bits out of old-style CAMGs modes before checking
 * availability. (some ILBM CAMG's have these bits set in old 1.3 modes,
 * and should not) If not an extended monitor ID, or if marked as
 * extended but missing upper 16 bits, screen out inappropriate bits now.
 */
if((!(modeid & MONITOR_ID_MASK)) ||
    ((modeid & EXTENDED_MODE)&&(!(modeid & 0xFFFF0000))))
        modeid &=
            (~(EXTENDED_MODE|SPRITES|GENLOCK_AUDIO|GENLOCK_VIDEO|VP_HIDE));

/* Check for bogus CAMG like some brushes have, with junk in
 * upper word and extended bit NOT set not set in lower word.
 */
if((modeid & 0xFFFF0000)&&(!(modeid & EXTENDED_MODE)))
    {
    /* Bad CAMG, so ignore CAMG and determine a mode based on
     * based on pagesize or aspect
     */
    modeid = NULL;
    if(wide >= 640)    modeid |= HIRES;
    if(high >= 400)    modeid |= LACE;
    }
```

The following example, camg.c, illustrates how to handle the possible CAMG
values.