

```

/* Camg.c - Execute me to compile me with SAS C 5.10
LC -bl -cfistq -v -y -j73 camg.c
Blink FROM LIB:c.o,camg.o TO camg LIBRARY LIB:LC.lib,LIB:Amiga.lib
quit
*/

/*
 * Camg.c Demonstrates use of ILBM.CAMG value. Filters out junk CAMGs and bad
 * bits. Adapts to 2.0 and 1.3
 *
 * Note - Under 2,0, when writing a CAMG you should write the 32-bit value
 * returned by the 2.0 graphics function GetVPMODEID(viewport);
 */

#include <exec/types.h>
#include <exec/libraries.h>
#include <intuition/intuition.h>
#include <graphics/displayinfo.h>

#include <clib/exec_protos.h>
#include <clib/intuition_protos.h>
#include <clib/graphics_protos.h>

#include <stdio.h>

struct Screen *openscreen(SHORT wide, SHORT high, SHORT deep, ULONG modeid);
ULONG modefallback(ULONG modeid, SHORT wide, SHORT high, SHORT deep);

#define DEBUG

struct Library *GfxBase;
struct Library *IntuitionBase;

void main(int argc, char **argv)
{
    struct Screen *screen;

    /*
     * Test the routines by passing various bad or possibly unavailable CAMG
     * modeid values
     */
    if (GfxBase = OpenLibrary("graphics.library", 0))
    {
        if (IntuitionBase = OpenLibrary("intuition.library", 0))
        {
            /* Junk CAMG */
            if (screen = openscreen(640, 400, 3, 0x12340000))
                CloseScreen(screen);

            /* Bad bits like VP_HIDE and SPRITES */
            if (screen = openscreen(320, 400, 3, 0x00006004))
                CloseScreen(screen);

            /* Extended bit set but not a valid modeid */
            if (screen = openscreen(320, 400, 3, 0x00009004))
                CloseScreen(screen);

            /* An ECS Mode - demonstrates modefallback if not available */
            if (screen = openscreen(640, 400, 2, 0x00039025))
                CloseScreen(screen);

            CloseLibrary(IntuitionBase);
        }
        CloseLibrary(GfxBase);
    }

    struct NewScreen ns = {NULL};

    /*
     * simple openscreen
     *
     * Passed width, height, depth, and modeid, attempts to open a 2.0 or 1.3 screen.
     * Demonstrates filtering out of bad CAMGs and modefallback() for unavailable
     * modes.
     */
}

```

```

struct Screen *
openscreen(SHORT wide, SHORT high, SHORT deep, ULONG modeid)
{
    extern struct Library *GfxBase;
    struct Screen *screen = NULL;

#ifdef DEBUG
    printf("\nAsked to open screen with modeid %$08lx\n", modeid);
#endif

    /*
     * BEFORE USING A CAMG YOU HAVE READ, YOU MUST SCREEN OUT JUNK CAMGs AND
     * INAPPROPRIATE BITS IN OLD 16-BIT CAMGs. YOU MAY WANT TO PLACE THESE
     * FIRST TWO TESTS WITH YOUR CAMG-READING CODE TO SCREEN OUT BAD CAMGs
     * IMMEDIATELY.
     */

    /*
     * Knock bad bits out of old-style CAMG 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;
        }

#ifdef DEBUG
    printf("After filtering, mode is now %$08lx\n", modeid);
#endif

    /*
     * Now see if mode is available
     */
    if (GfxBase->lib_Version >= 36)
    {
        /* if mode is not available, try a fallback mode */
        if (ModeNotAvailable(modeid))
            modeid = modefallback(modeid, wide, high, deep);

        if (!(ModeNotAvailable(modeid)) /* if mode is available */
            {
                /*
                 * We have an available mode id Here you may wish to create a
                 * custom, or centered, or overscan display clip based on the size
                 * of the image. Or just use one of the standard clips.
                 *
                 * The 2.0 Display program uses QueryOverscan to get the settings of
                 * this modeid's OSCAN_TEXT, OSCAN_STANDARD, and OSCAN_MAX. Display
                 * centers the screen (via TopEdge and LeftEdge) within the user's
                 * closest OSCAN rectangle, and creates a display clip by using the
                 * same values and then clipping the values to be within OSCAN_MAX
                 * limits. If the centered screen ends up lower than user's
                 * OSCAN_TEXT settings, I move it up to same MinY as his OSCAN_TEXT
                 * --- otherwise his Workbench might peek over the top.
                 */
            }
        }
    }
}

```

```

        /*
        * Now use extended OpenScreen or OpenScreenTags for this modeid.
        * (this gives you the benefit of system-supported overscan, etc.)
        */
#ifdef DEBUG
    printf("Doing OpenScreenTags of modeid %08lx\n", modeid);
#endif
    screen = OpenScreenTags(NULL,
        SA_Width, wide,
        SA_Height, high,
        SA_Depth, deep,
        SA_DisplayID, modeid);
}

if (!screen) /* not 2.0, or mode not
            * available, or can't open */
{
    /*
    * Try an old-style OpenScreen with modefallback() modeid
    */
    modeid = modefallback(modeid, wide, high, deep);

    ns.Width = wide;
    ns.Height = high;
    ns.Depth = deep;
    ns.ViewModes = modeid;
}

#ifdef DEBUG
    printf("Doing old OpenScreen of mode %04lx\n", modeid);
#endif
    screen = OpenScreen(&ns);
}

return (screen);
}

/*
 * modefallback - passed an unavailable modeid, attempts to provide an
 * available replacement modeid to use instead
 */
ULONG
modefallback(ULONG modeid, SHORT wide, SHORT high, SHORT deep)
{
    extern struct Library *GfxBase;
    ULONG newmodeid;

    /*
    * Here you should either be asking the user what mode they want OR
    * searching the display database and choosing an appropriate replacement
    * mode based on what you or the user deem important (colors, or aspect, or
    * size, etc.). You could also use a built in table for modes you know
    * about, and substitute mode you wish to use when the desired mode is not
    * available.
    *
    * This simple routine simply masks out everything but old modes. This is a
    * way to get some kind of display open but not necessarily a suitable or
    * compatible display.
    */

    newmodeid = modeid & (HIRES | LACE | HAM | EXTRA_HALFBRITE);

#ifdef DEBUG
    printf("Modefallback passed 0x%08lx, returning 0x%08lx\n", modeid, newmodeid);
#endif
    return (newmodeid);
}

```

