Abstract

This document describes the graphics driver format used for DJGPP and the LIBGRX graphics library. It also gives hints for creating a driver for an unsupported display adapter.

Introduction

The DJGPP graphics drivers do two things:

(1) Invoke the BIOS INT 10 routine for setting up the desired graphics mode. Different boards support different resolutions and use different mode numbers -- that's why different drivers are needed for different boards.

(2) Implement page mapping for video modes which use more than 64K of video memory. This is again board dependent.

Old driver format

The following C declarations describe the header of an old format DJGPP graphics driver. Of course, real drivers are coded in assembly.

```
typedef unsigned short u_short;
typedef unsigned char  u_char;

struct old_driver_header {
    u_short      mode_set_routine_offset;
    u_short      paging_routine_offset;
    u_short      paging_mode_flag;                        /* 0 if no separate R/W, 1 if yes */
    u_short      default_text_width;
    u_short      default_text_height;
    u_short      default_graphics_width;
    u_short      default_graphics_height;
};
```

The mode set routine does the following:

```
;--------------------------------------------------------------------------
; Entry: AX=mode selection
;               0=80x25 text
;               1=default text
;               2=text CX cols by DX rows
;               3=biggest text
;               4=320x200 graphics
;               5=default graphics
;               6=graphics CX width by DX height
;               7=biggest non-interlaced graphics
;               8=biggest graphics
;        CX=width (in pixels or characters) (not always used -- depends on AX)
;        DX=height
;
; NOTE: This runs in real mode, but don't mess with the segment registers.
;
; Exit:  CX=width (in pixels or characters)
;        DX=height
;--------------------------------------------------------------------------
```

The paging routine does the following:

```
;--------------------------------------------------------------------------
; Entry: AH=read page
;        AL=write page
;
; NOTE: This runs in protected mode!  Don't mess with the segment registers!
; This code must be relocatable and may not reference any data!
;
; Exit: VGA configured.
;       AX,BX,CX,DX,SI,DI may be trashed
;--------------------------------------------------------------------------
```

The old style graphics driver structure remained unchanged for the first 16 color drivers developed for LIBGRX. The only difference is that the additional 15 bits in the third word of the header were given new meanings. (The 256 color DJGPP library only used one bit to encode the capability to map different pages for reading and writing.) The values of these new bitfields were assigned as to stay compatible with the existing 256 color drivers. (I.e. the 0 value in every bitfield selects the 256 color VGA option.) The definition of these bits (from "grdriver.h"):

```
#define GRD_NEW_DRIVER  0x0008            /* NEW FORMAT DRIVER IF THIS IS SET */

#define GRD_PAGING_MASK  0x0007           /* mask for paging modes */
#define GRD_NO_RW        0x0000           /* standard paging, no separate R/W */
#define GRD_RW_64K       0x0001           /* two separate 64K R/W pages */
/* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! */
/* THE FOLLOWING THREE OPTIONS ARE NOT SUPPORTED AT THIS TIME         */
#define GRD_RW_32K       0x0002           /* two separate 32Kb pages */
#define GRD_MAP_128K     0x0003           /* 128Kb memory map -- some Tridents
                                             can do it (1024x768x16 without
                                             paging!!!)
#define GRD_MAP_EXTMEM   0x0004           /* Can be mapped extended, above 1M.
                                             Some Tseng 4000-s can do it, NO
                                             PAGING AT ALL!!!! */
/* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! */

#define GRD_TYPE_MASK    0xf000           /* adapter type mask */
#define GRD_VGA          0x0000           /* vga */
#define GRD_EGA          0x1000           /* ega */
#define GRD_HERC         0x2000           /* hercules */
#define GRD_8514A        0x3000           /* 8514/A or compatible */
#define GRD_S3           0x4000           /* S3 graphics accelerator */

#define GRD_PLANE_MASK   0x0f00           /* bitplane number mask */
#define GRD_8_PLANES     0x0000           /* 8 planes = 256 colors */
#define GRD_4_PLANES     0x0100           /* 4 planes = 16 colors */
#define GRD_1_PLANE      0x0200           /* 1 plane  = 2 colors */
#define GRD_16_PLANES    0x0300           /* VGA with 32K colors */
#define GRD_8_X_PLANES   0x0400           /* VGA in mode X w/ 256 colors */

#define GRD_MEM_MASK     0x00f0           /* memory size mask */
#define GRD_64K          0x0010           /* 64K display memory */
#define GRD_128K         0x0020           /* 128K display memory */
#define GRD_256K         0x0030           /* 256K display memory */
#define GRD_512K         0x0040           /* 512K display memory */
#define GRD_1024K        0x0050           /* 1MB display memory */
#define GRD_192K         0x0060           /* 192K -- some 640x480 EGA-s */
#define GRD_M_NOTSPEC    0x0000           /* memory amount not specified */
```

An old style driver has the 'GRD_NEW_DRIVER' bit cleared. It can work with previous versions of GO32. Of course for 16 color support the application has to be linked with the LIBGRX library instead of the original 256 color library.

The following additional old format graphics drivers are supplied with the LIBGRX graphics library:

EGA16.GRD	16 color EGA driver (640x350x16 max. resolution)
VGA16.GRD	16 color standard VGA driver (640x480x16 max. resolution)
TSENG4KN.GRD	same as DJGPP's Tseng ET 4000 256 color driver, but with added support for the 100x40 text mode. (max: 1024x768x256)
TSENG416.GRD	Tseng ET 4000 16 color driver. (max: 1024x768x16)
TRID89N.GRD	Trident 8900 256 color driver. This driver has an updated paging routine which seems to fix some previous problems on boards with recent enough chipsets. (max: 1024x768x256)
TRID8916.GRD:	Trident 8900 16 color driver (max: 1024x768x16)

New driver format

The disadvantage of the old driver format is that the number of colors is not programmable. The new driver format solves this problem and it also gives the application program a way to query the driver for a list of the supported text and graphics modes. For this the driver header was extended as follows:

```
struct new_driver_header {
    u_short      mode_set_routine_offset;
    u_short      paging_routine_offset;
```

```
    u_short      driver_mode_flag;                /* flag word, see bits below */
    u_short      default_text_width;
    u_short      default_text_height;
    u_short      default_graphics_width;
    u_short      default_graphics_height;
    u_short      default_color_number;            /* NEW, may be set from environment */
    u_short      init_routine_offset;             /* NEW, call once after drvr loaded */
    u_short      text_mode_table_offset;          /* NEW, table of supported text modes */
    u_short      graphics_mode_table_offset;      /* NEW, table of supported graphics modes */
};
```

'text_mode_table_offset' points to a table with entries as follows:

```
struct text_mode_entry {
    u_short      columns;
    u_short      rows;
    u_short      number_of_colors;                /* in text mode it is mainly here to make
                                                     GCC happy with the alignments */
    u_char       BIOS_mode;                       /* BIOS mode number. Mode not available on
                                                     the current card if this field is 0xff. */
    u_char       special;                         /* if non zero then the driver does some
                                                     special hacks to set it up (like
                                                     the 50 row mode on a standard VGA) */
};
```

The end of the table is marked by an all 0 entry. The table entries are sorted by increasing size. The field 'graphics_mode_table_offset' points to a table with entries as follows:

```
struct graphics_mode_entry {
    u_short      width;
    u_short      height;
    u_short      number_of_colors;
    u_char       BIOS_mode;                       /* BIOS mode number. Mode not available on
                                                     the current card if this field is 0xff.
                                                     (This may happen for example if the card
                                                     is not fully populated with RAM) */
    u_char       special;                         /* if non zero then the driver does some
                                                     special hacks to set it up (like
                                                     setting up the 32768 color mode on
                                                     some ET4000 cards) */
};
```

The end of the table is marked by an all 0 entry. The table is sorted by increasing color number and within the same color modes by increasing size.

If the driver is of the new type then it also supports an initialization routine. This is called once after the driver is loaded. The initialization routine can do one or more of the following:

(1) Check whether the video card is of the expected type

(2) Determine the amount of video memory on board.

(3) Determine which of the modes in the text and graphics mode tables are actually available (due to video RAM and possibly DAC [32768 colors!!] limitations) and mark the unavailable entries in the tables.

To use the new format drivers a recent version of GO32 (1.06, after the middle of April 1992) has to be used. Such versions should recognize the "nc <number>" option field in the GO32 environment variable which specifies the default number of colors for the driver.

The following new format drivers have been included with the LIBGRX library (new drivers have the ".GRN" extension):

STDEGA.GRN	standard EGA 16 color driver
STDVGA.GRN	standard VGA 16 and 256 color driver
ET4000.GRN	Tseng ET 4000 16 256 and 32768 color driver
TR8900.GRN	Trident 8900 16 and 256 color driver
ATIULTRA.GRN	Driver for the ATI ULTRA board. This board actually contains two graphics

adapters: a 512 KByte SVGA board and a 8514/A compatible accelerator. The driver was programmed to use the VGA part for text, 16 color graphics, and low resolution 256 color graphics modes and the 8514/A part for high resolution 256 color modes. This driver should work with any VGA+8514/A (the 8514/A MUST have 1MB memory for 1024x768 resolution) combination as long as the ATI-specific modes of the VGA part are not used.

STEALTH.GRN Driver for the Diamond Stealth S3 graphics accelerator board. The driver was programmed to use VGA-style video RAM access for text, 16 color graphics, and low resolution 256 color graphics modes and the S3 accelarator functions for high resolution 256 color modes. Currently no 32768 color modes are supported on S3 boards. This driver should work with other S3-based boards which have a VESA BIOS.

Creating a driver for an unsupported board

You can only use EGA or VGA boards in 16 256 or 32768 color modes with the graphics library. In the near future there will be support for Hercules boards as well. SUPPORT IS NOT PLANNED FOR: BOARDS WITH ON-BOARD GRAPHICS PROCESSORS (TIGA, 8514A, etc...) To create a driver for an unsupported EGA or VGA board you will need the followings:

(1) An assembler (TASM or MASM), a linker (TLINK or LINK) and a utility to convert .EXE files to the .COM format (EXE2BIN). See also the ’makefile’ in the ’drivers’ sub-directory.
(2) A documentation of the board containing the supported BIOS mode numbers and resolutions.
(3) If the driver needs to support modes which use a memory map bigger than 64K then you also need a piece of code which does page switching on your board. (Any mode above 800x600 in 16 colors or 320x200 in 256 colors DOES USE paging.) Possible sources:
    - a working, tested 256 color original DJGPP driver (if you just want to convert it to the new format).
    - VGAKIT.ZIP (available from various archive sites, like wuarchive, simtel, etc...)
    - various books on the subject.

It is best to start with the source of a driver supporting similar resolutions as your board. Use the proper format driver, i.e. if you want a new format driver start with a new original, etc... Typically the driver sources are relatively well commented. What you need to do is:

(1) possibly change the option word at the head of the driver to indicate the number of colors (old format only), the amount of memory on board, the memory mapping capabilities of the board.
(2) change the mode setting code to use the proper BIOS mode numbers. In the old format drivers these are somewhat scattered throughout the code, in the new format drivers you need to edit a few tables only.
(3) Replace the paging routine with the one suitable for your board. If your driver supports 16 color resolutions beyond 800x600 then you have to make sure that upon exit from the paging routine the graphics controller port’s (0x3ce) index register is reset to 8. (See the paging routine in "TR8900.ASM".) If the paging mechanism does not use any register accessed through the graphics controller port, then you don’t need to worry about this.