Video Mode Selection Support 2.13

Intro

This small document describes the "Video Mode Selection" feature which allows the use of various special video modes supported by the video BIOS. Due to usage of the BIOS, the selection is limited to boot time (before the kernel decompression starts) and works only on 80X86 machines that are booted through BIOS firmware (as opposed to through UEFI, kexec, etc.).

> **Note**
>
> Short intro for the impatient: Just use vga=ask for the first time, enter `scan` on the video mode prompt, pick the mode you want to use, remember its mode ID (the four-digit hexadecimal number) and then set the vga parameter to this number (converted to decimal first).

The video mode to be used is selected by a kernel parameter which can be specified in the kernel Makefile (the SVGA_MODE=... line) or by the "vga=..." option of LILO (or some other boot loader you use) or by the "xrandr" utility (present in standard Linux utility packages). You can use the following values of this parameter:

```
NORMAL_VGA - Standard 80x25 mode available on all display adapters.

EXTENDED_VGA - Standard 8-pixel font mode: 80x43 on EGA, 80x50 on VGA.

ASK_VGA - Display a video mode menu upon startup (see below).

0..35 - Menu item number (when you have used the menu to view the list of
   modes available on your adapter, you can specify the menu item you want
   to use). 0..9 correspond to "0".."9", 10..35 to "a".."z". Warning: the
   mode list displayed may vary as the kernel version changes, because the
   modes are listed in a "first detected -- first displayed" manner. It's
   better to use absolute mode numbers instead.

0x.... - Hexadecimal video mode ID (also displayed on the menu, see below
   for exact meaning of the ID). Warning: LILO doesn't support
   hexadecimal numbers -- you have to convert it to decimal manually.
```

Menu

The ASK_VGA mode causes the kernel to offer a video mode menu upon bootup. It displays a "Press <RETURN> to see video modes available, <SPACE> to continue or wait 30 secs" message. If you press <RETURN>, you enter the menu, if you press <SPACE> or wait 30 seconds, the kernel will boot up in the standard 80x25 mode.

The menu looks like:

```
Video adapter: <name-of-detected-video-adapter>
Mode:    COLSxROWS:
0  0F00  80x25
1  0F01  80x50
2  0F02  80x43
3  0F03  80x26
....
Enter mode number or ``scan``: <flashing-cursor-here>
```

<name-of-detected-video-adapter> tells what video adapter did Linux detect -- it's either a generic adapter name (MDA, CGA, HGC, EGA, VGA, VESA VGA [a VGA with VESA-compliant BIOS]) or a chipset name (e.g., Trident). Direct detection of chipsets is turned off by default as it's inherently unreliable due to absolutely insane PC design.

"0 0F00 80x25" means that the first menu item (the menu items are numbered from "0" to "9" and from "a" to "z") is a 80x25 mode with ID=0x0f00 (see the next section for a description of mode IDs).

<flashing-cursor-here> encourages you to enter the item number or mode ID you wish to set and press <RETURN>. If the computer complains something about "Unknown mode ID", it is trying to tell you that it isn't possible to set such a mode. It's also possible to press only <RETURN> which leaves the current mode.

The mode list usually contains a few basic modes and some VESA modes. In case your chipset has been detected, some chipset-specific modes are shown as well (some of these might be missing or unusable on your machine as different BIOSes are often shipped with the same card and the mode numbers depend purely on the VGA BIOS).

The modes displayed on the menu are partially sorted: The list starts with the standard modes (80x25 and 80x50) followed by "special" modes (80x28 and 80x43), local modes (if the local modes feature is enabled), VESA modes and finally SVGA modes for the auto-detected adapter.

If you are not happy with the mode list offered (e.g., if you think your card is able to do more), you can enter "scan" instead of item number / mode ID. The program will try to ask the BIOS for all possible video mode numbers and test what happens then. The screen will be probably flashing wildly for some time and strange noises will be heard from inside the monitor and so on and then, really all consistent video modes supported by your BIOS will appear (plus maybe some `ghost modes`). If you are afraid this could damage your monitor, don't use this function.

After scanning, the mode ordering is a bit different: the auto-detected SVGA modes are not listed at all and the modes revealed by `scan` are shown before all VESA modes.

Mode IDs

Because of the complexity of all the video stuff, the video mode IDs used here are also a bit complex. A video mode ID is a 16-bit number usually expressed in a hexadecimal notation (starting with "0x"). You can set a mode by entering its mode directly if you know it even if it isn't shown on the menu.

The ID numbers can be divided to those regions:

```
0x0000 to 0x00ff - menu item references. 0x0000 is the first item. Don't use
     outside the menu as this can change from boot to boot (especially if you
     have used the ``scan`` feature).

0x0100 to 0x017f - standard BIOS modes. The ID is a BIOS video mode number
     (as presented to INT 10, function 00) increased by 0x0100.

0x0200 to 0x08ff - VESA BIOS modes. The ID is a VESA mode ID increased by
     0x0100. All VESA modes should be autodetected and shown on the menu.

0x0900 to 0x09ff - Video7 special modes. Set by calling INT 0x10, AX=0x6f05.
     (Usually 940=80x43, 941=132x25, 942=132x44, 943=80x60, 944=100x60,
     945=132x28 for the standard Video7 BIOS)

0x0f00 to 0x0fff - special modes (they are set by various tricks -- usually
     by modifying one of the standard modes). Currently available:
     0x0f00  standard 80x25, don't reset mode if already set (=FFFF)
     0x0f01  standard with 8-point font: 80x43 on EGA, 80x50 on VGA
     0x0f02  VGA 80x43 (VGA switched to 350 scanlines with a 8-point font)
     0x0f03  VGA 80x28 (standard VGA scans, but 14-point font)
     0x0f04  leave current video mode
     0x0f05  VGA 80x30 (480 scans, 16-point font)
     0x0f06  VGA 80x34 (480 scans, 14-point font)
     0x0f07  VGA 80x60 (480 scans, 8-point font)
     0x0f08  Graphics hack (see the VIDEO_GFX_HACK paragraph below)

0x1000 to 0x7fff - modes specified by resolution. The code has a "0xRRCC"
     form where RR is a number of rows and CC is a number of columns.
     E.g., 0x1950 corresponds to a 80x25 mode, 0x2b84 to 132x43 etc.
     This is the only fully portable way to refer to a non-standard mode,
     but it relies on the mode being found and displayed on the menu
     (remember that mode scanning is not done automatically).

0xff00 to 0xffff - aliases for backward compatibility:
     0xffff  equivalent to 0x0f00 (standard 80x25)
     0xfffe  equivalent to 0x0f01 (EGA 80x43 or VGA 80x50)
```

If you add 0x8000 to the mode ID, the program will try to recalculate vertical display timing according to mode parameters, which can be used to eliminate some annoying bugs of certain VGA BIOSes (usually those used for cards with S3 chipsets and old Cirrus Logic BIOSes) -- mainly extra lines at the end of the display.

Options

Build options for arch/x86/boot/* are selected by the kernel kconfig utility and the kernel .config file.

VIDEO_GFX_HACK - includes special hack for setting of graphics modes to be used later by special drivers. Allows to set _any_ BIOS mode including graphic ones and forcing specific text screen resolution instead of peeking it from BIOS variables. Don't use unless you think you know what you're doing. To activate this setup, use mode number 0x0f08 (see the Mode IDs section above).

Still doesn't work?

When the mode detection doesn't work (e.g., the mode list is incorrect or the machine hangs instead of displaying the menu), try to switch off some of the configuration options listed under "Options". If it fails, you can still use your kernel with the video mode set directly via the kernel parameter.

In either case, please send me a bug report containing what _exactly_ happens and how do the configuration switches affect the behaviour of the bug.

If you start Linux from M$-DOS, you might also use some DOS tools for video mode setting. In this case, you must specify the

0x0f04 mode ("leave current settings") to Linux, because if you don't and you use any non-standard mode, Linux will switch to 80x25 automatically.

If you set some extended mode and there's one or more extra lines on the bottom of the display containing already scrolled-out text, your VGA BIOS contains the most common video BIOS bug called "incorrect vertical display end setting". Adding 0x8000 to the mode ID might fix the problem. Unfortunately, this must be done manually -- no autodetection mechanisms are available.

History

1.0 (??-Nov-95)	First version supporting all adapters supported by the old setup.S + Cirrus Logic 54XX. Present in some 1.3.4? kernels and then removed due to instability on some machines.
2.0 (28-Jan-96)	Rewritten from scratch. Cirrus Logic 64XX support added, almost everything is configurable, the VESA support should be much more stable, explicit mode numbering allowed, "scan" implemented etc.
2.1 (30-Jan-96)	VESA modes moved to 0x200-0x3ff. Mode selection by resolution supported. Few bugs fixed. VESA modes are listed prior to modes supplied by SVGA autodetection as they are more reliable. CLGD autodetect works better. Doesn't depend on 80x25 being active when started. Scanning fixed. 80x43 (any VGA) added. Code cleaned up.
2.2 (01-Feb-96)	EGA 80x43 fixed. VESA extended to 0x200-0x4ff (non-standard 02XX VESA modes work now). Display end bug workaround supported. Special modes renumbered to allow adding of the "recalculate" flag, 0xffff and 0xfffe became aliases instead of real IDs. Screen contents retained during mode changes.
2.3 (15-Mar-96)	Changed to work with 1.3.74 kernel.
2.4 (18-Mar-96)	Added patches by Hans Lermen fixing a memory overwrite problem with some boot loaders. Memory management rewritten to reflect these changes. Unfortunately, screen contents retaining works only with some loaders now. Added a Tseng 132x60 mode.
2.5 (19-Mar-96)	Fixed a VESA mode scanning bug introduced in 2.4.
2.6 (25-Mar-96)	Some VESA BIOS errors not reported -- it fixes error reports on several cards with broken VESA code (e.g., ATI VGA).
2.7 (09-Apr-96)	• Accepted all VESA modes in range 0x100 to 0x7ff, because some cards use very strange mode numbers. • Added Realtek VGA modes (thanks to Gonzalo Tornaria). • Hardware testing order slightly changed, tests based on ROM contents done as first. • Added support for special Video7 mode switching functions (thanks to Tom Vander Aa). • Added 480-scanline modes (especially useful for notebooks, original version written by hhanemaa@cs.ruu.nl, patched by Jeff Chua, rewritten by me). • Screen store/restore fixed.
2.8 (14-Apr-96)	• Previous release was not compilable without CONFIG_VIDEO_SVGA. • Better recognition of text modes during mode scan.
2.9 (12-May-96)	• Ignored VESA modes 0x80 - 0xff (more VESA BIOS bugs!)
2.10(11-Nov-96)	• The whole thing made optional. • Added the CONFIG_VIDEO_400_HACK switch. • Added the CONFIG_VIDEO_GFX_HACK switch. • Code cleanup.
2.11(03-May-97)	• Yet another cleanup, now including also the documentation. • Direct testing of SVGA adapters turned off by default, `scan` offered explicitly on the prompt line. • Removed the doc section describing adding of new probing functions as I try to get rid of _all_ hardware probing here.
2.12(25-May-98)	Added support for VESA frame buffer graphics.
2.13(14-May-99)	Minor documentation fixes.