What is matroxfb?

This is a driver for a graphic framebuffer for Matrox devices on Alpha, Intel and PPC boxes.

Advantages:

- It provides a nice large console (128 cols + 48 lines with 1024x768) without using tiny, unreadable fonts.
- You can run XF{68,86}_FBDev or XFree86 fbdev driver on top of /dev/fb0
- Most important: boot logo :-)

Disadvantages:

- graphic mode is slower than text mode... but you should not notice if you use same resolution as you used in textmode.

How to use it?

Switching modes is done using the video=matroxfb:vesa:... boot parameter or using *fbset* program.

If you want, for example, enable a resolution of 1280x1024x24bpp you should pass to the kernel this command line: "video=matroxfb:vesa:0x1BB".

You should compile in both vgacon (to boot if you remove you Matrox from box) and matroxfb (for graphics mode). You should not compile-in vesafb unless you have primary display on non-Matrox VBE2.0 device (see Documentation/fb/vesafb.rst for details).

Currently supported video modes are (through vesa:... interface, PowerMac has [as addon] compatibility code):

Graphic modes

bpp	640x400	640x480	768x576	800x600	960x720
4		0x12		0x102	
8	0x100	0x101	0x180	0x103	0x188
15		0x110	0x181	0x113	0x189
16		0x111	0x182	0x114	0x18A
24		0x1B2	0x184	0x1B5	0x18C
32		0x112	0x183	0x115	0x18B

Graphic modes (continued)

bpp	1024x768	1152x864	1280x1024	1408x1056	1600x1200
4	0x104		0x106		
8	0x105	0x190	0x107	0x198	0x11C
15	0x116	0x191	0x119	0x199	0x11D
16	0x117	0x192	0x11A	0x19A	0x11E
24	0x1B8	0x194	0x1BB	0x19C	0x1BF
32	0x118	0x193	0x11B	0x19B	

Text modes

text	640x400	640x480	1056x344	1056x400	1056x480
8x8	0x1C0	0x108	0x10A	0x10B	0x10C
8x16	2, 3, 7			0x109	

You can enter these number either hexadecimal (leading *0x*) or decimal (0x100 = 256). You can also use value + 512 to achieve compatibility with your old number passed to vesafb.

Non-listed number can be achieved by more complicated command-line, for example 1600x1200x32bpp can be specified by *video=matroxfb:vesa:0x11C,depth:32*.

X11

XF{68,86}_FBDev should work just fine, but it is non-accelerated. On non-intel architectures there are some glitches for 24bpp videomodes. 8, 16 and 32bpp works fine.

Running another (accelerated) X-Server like XF86_SVGA works too. But (at least) XFree servers have big troubles in multihead configurations (even on first head, not even talking about second). Running XFree86 4.x accelerated mga driver is possible, but you must not enable DRI - if you do, resolution and color depth of your X desktop must match resolution and color depths of your virtual consoles, otherwise X will corrupt accelerator settings.

SVGALib

Driver contains SVGALib compatibility code. It is turned on by choosing textual mode for console. You can do it at boot time by using videomode 2,3,7,0x108-0x10C or 0x1C0. At runtime, *fbset -depth 0* does this work. Unfortunately, after SVGALib application exits, screen contents is corrupted. Switching to another console and back fixes it. I hope that it is SVGALib's problem and not mine, but I'm not sure.

Configuration

You can pass kernel command line options to matroxfb with *video=matroxfb:option1,option2:value2,option3* (multiple options should be separated by comma, values are separated from options by *:*). Accepted options:

mem:X	size of memory (X can be in megabytes, kilobytes or bytes) You can only decrease value determined by driver because of it always probe for memory. Default is to use whole detected memory usable for on-screen display (i.e. max. 8 MB).
disabled	do not load driver; you can use also *off*, but *disabled* is here too.
enabled	load driver, if you have *video=matroxfb:disabled* in LILO configuration, you can override it by this (you cannot override *off*). It is default.
noaccel	do not use acceleration engine. It does not work on Alphas.
accel	use acceleration engine. It is default.
nopan	create initial consoles with vyres = yres, thus disabling virtual scrolling.
pan	create initial consoles as tall as possible (vyres = memory/vxres). It is default.
nopciretry	disable PCI retries. It is needed for some broken chipsets, it is autodetected for intel's 82437. In this case device does not comply to PCI 2.1 specs (it will not guarantee that every transaction terminate with success or retry in 32 PCLK).
pciretry	enable PCI retries. It is default, except for intel's 82437.
novga	disables VGA I/O ports. It is default if BIOS did not enable device. You should not use this option, some boards then do not restart without power off.
vga	preserve state of VGA I/O ports. It is default. Driver does not enable VGA I/O if BIOS did not it (it is not safe to enable it in most cases).
nobios	disables BIOS ROM. It is default if BIOS did not enable BIOS itself. You should not use this option, some boards then do not restart without power off.
bios	preserve state of BIOS ROM. It is default. Driver does not enable BIOS if BIOS was not enabled before.
noinit	tells driver, that devices were already initialized. You should use it if you have G100 and/or if driver cannot detect memory, you see strange pattern on screen and so on. Devices not enabled by BIOS are still initialized. It is default.
init	driver initializes every device it knows about.
memtype	specifies memory type, implies 'init'. This is valid only for G200 and G400 and has following meaning: G200: • 0 -> 2x128Kx32 chips, 2MB onboard, probably sgram • 1 -> 2x128Kx32 chips, 4MB onboard, probably sgram • 2 -> 2x256Kx32 chips, 4MB onboard, probably sgram • 3 -> 2x256Kx32 chips, 8MB onboard, probably sgram • 4 -> 2x512Kx16 chips, 8/16MB onboard, probably sdram only • 5 -> same as above • 6 -> 4x128Kx32 chips, 4MB onboard, probably sgram • 7 -> 4x128Kx32 chips, 8MB onboard, probably sgram G400: • 0 -> 2x512Kx16 SDRAM, 16/32MB • 2x512Kx32 SGRAM, 16/32MB • 1 -> 2x256Kx32 SGRAM, 8/16MB • 2 -> 4x128Kx32 SGRAM, 8/16MB • 3 -> 4x512Kx32 SDRAM, 32MB • 4 -> 4x256Kx32 SGRAM, 16/32MB • 5 -> 2x1Mx32 SDRAM, 32MB • 6 -> reserved • 7 -> reserved You should use sdram or sgram parameter in addition to memtype parameter.
nomtrr	disables write combining on frame buffer. This slows down driver but there is reported minor incompatibility between GUS DMA and XFree under high loads if write combining is enabled (sound dropouts).
mtrr	enables write combining on frame buffer. It speeds up video accesses much. It is default. You must have MTRR support enabled in kernel and your CPU must have MTRR (f.e. Pentium II have them).

sgram	tells to driver that you have Gxx0 with SGRAM memory. It has no effect without *init*.
sdram	tells to driver that you have Gxx0 with SDRAM memory. It is a default.
inv24	change timings parameters for 24bpp modes on Millennium and Millennium II. Specify this if you see strange color shadows around characters.
noinv24	use standard timings. It is the default.
inverse	invert colors on screen (for LCD displays)
noinverse	show true colors on screen. It is default.
dev:X	bind driver to device X. Driver numbers device from 0 up to N, where device 0 is first *known* device found, 1 second and so on. lspci lists devices in this order. Default is *every* known device.
nohwcursor	disables hardware cursor (use software cursor instead).
hwcursor	enables hardware cursor. It is default. If you are using non-accelerated mode (*noaccel* or *fbset -accel false*), software cursor is used (except for text mode).
noblink	disables cursor blinking. Cursor in text mode always blinks (hw limitation).
blink	enables cursor blinking. It is default.
nofastfont	disables fastfont feature. It is default.
fastfont:X	enables fastfont feature. X specifies size of memory reserved for font data, it must be >= (fontwidth*fontheight*chars_in_font)/8. It is faster on Gx00 series, but slower on older cards.
grayscale	enable grayscale summing. It works in PSEUDOCOLOR modes (text, 4bpp, 8bpp). In DIRECTCOLOR modes it is limited to characters displayed through putc/putcs. Direct accesses to framebuffer can paint colors.
nograyscale	disable grayscale summing. It is default.
cross4MB	enables that pixel line can cross 4MB boundary. It is default for non-Millennium.
nocross4MB	pixel line must not cross 4MB boundary. It is default for Millennium I or II, because of these devices have hardware limitations which do not allow this. But this option is incompatible with some (if not all yet released) versions of XF86_FBDev.
dfp	enables digital flat panel interface. This option is incompatible with secondary (TV) output - if DFP is active, TV output must be inactive and vice versa. DFP always uses same timing as primary (monitor) output.
dfp:X	use settings X for digital flat panel interface. X is number from 0 to 0xFF, and meaning of each individual bit is described in G400 manual, in description of DAC register 0x1F. For normal operation you should set all bits to zero, except lowest bit. This lowest bit selects who is source of display clocks, whether G400, or panel. Default value is now read back from hardware - so you should specify this value only if you are also using *init* parameter.
outputs:XYZ	set mapping between CRTC and outputs. Each letter can have value of 0 (for no CRTC), 1 (CRTC1) or 2 (CRTC2), and first letter corresponds to primary analog output, second letter to the secondary analog output and third letter to the DVI output. Default setting is 100 for cards below G400 or G400 without DFP, 101 for G400 with DFP, and 111 for G450 and G550. You can set mapping only on first card, use matroxset for setting up other devices.
vesa:X	selects startup videomode. X is number from 0 to 0x1FF, see table above for detailed explanation. Default is 640x480x8bpp if driver has 8bpp support. Otherwise first available of 640x350x4bpp, 640x480x15bpp, 640x480x24bpp, 640x480x32bpp or 80x25 text (80x25 text is always available).

If you are not satisfied with videomode selected by *vesa* option, you can modify it with these options:

xres:X	horizontal resolution, in pixels. Default is derived from *vesa* option.
yres:X	vertical resolution, in pixel lines. Default is derived from *vesa* option.
upper:X	top boundary: lines between end of VSYNC pulse and start of first pixel line of picture. Default is derived from *vesa* option.
lower:X	bottom boundary: lines between end of picture and start of VSYNC pulse. Default is derived from *vesa* option.
vslen:X	length of VSYNC pulse, in lines. Default is derived from *vesa* option.
left:X	left boundary: pixels between end of HSYNC pulse and first pixel. Default is derived from *vesa* option.
right:X	right boundary: pixels between end of picture and start of HSYNC pulse. Default is derived from *vesa* option.
hslen:X	length of HSYNC pulse, in pixels. Default is derived from *vesa* option.
pixclock:X	dotclocks, in ps (picoseconds). Default is derived from *vesa* option and from *fh* and *fv* options.
sync:X	sync. pulse - bit 0 inverts HSYNC polarity, bit 1 VSYNC polarity. If bit 3 (value 0x08) is set, composite sync instead of HSYNC is generated. If bit 5 (value 0x20) is set, sync on green is turned on. Do not forget that if you want sync on green, you also probably want composite sync. Default depends on *vesa*.
depth:X	Bits per pixel: 0=text, 4,8,15,16,24 or 32. Default depends on *vesa*.

If you know capabilities of your monitor, you can specify some (or all) of *maxclk*, *fh* and *fv*. In this case, *pixclock* is computed so that pixclock <= maxclk, real_fh <= fh and real_fv <= fv.

maxclk:X	maximum dotclock. X can be specified in MHz, kHz or Hz. Default is *don`t care*.
fh:X	maximum horizontal synchronization frequency. X can be specified in kHz or Hz. Default is *don't care*.
fv:X	maximum vertical frequency. X must be specified in Hz. Default is 70 for modes derived from *vesa* with yres <= 400, 60Hz for yres > 400.

Limitations

Limitations

There are known and unknown bugs, features and misfeatures. Currently there are following known bugs:

- SVGALib does not restore screen on exit
- generic fbcon-cfbX procedures do not work on Alphas. Due to this, *noaccel* (and cfb4 accel) driver does not work on Alpha. So everyone with access to */dev/fb** on Alpha can hang machine (you should restrict access to */dev/fb** - everyone with access to this device can destroy your monitor, believe me...).
- 24bpp does not support correctly XF-FBDev on big-endian architectures.
- interlaced text mode is not supported; it looks like hardware limitation, but I'm not sure.
- Gxx0 SGRAM/SDRAM is not autodetected.
- maybe more...

And following misfeatures:

- SVGALib does not restore screen on exit.
- pixclock for text modes is limited by hardware to
    - 83 MHz on G200
    - 66 MHz on Millennium I
    - 60 MHz on Millennium II

  Because I have no access to other devices, I do not know specific frequencies for them. So driver does not check this and allows you to set frequency higher that this. It causes sparks, black holes and other pretty effects on screen. Device was not destroyed during tests. :-)
- my Millennium G200 oscillator has frequency range from 35 MHz to 380 MHz (and it works with 8bpp on about 320 MHz dotclocks (and changed mclk)). But Matrox says on product sheet that VCO limit is 50-250 MHz, so I believe them (maybe that chip overheats, but it has a very big cooler (G100 has none), so it should work).
- special mixed video/graphics videomodes of Mystique and Gx00 - 2G8V16 and G16V16 are not supported
- color keying is not supported
- feature connector of Mystique and Gx00 is set to VGA mode (it is disabled by BIOS)
- DDC (monitor detection) is supported through dualhead driver
- some check for input values are not so strict how it should be (you can specify vslen=4000 and so on).
- maybe more...

And following features:

- 4bpp is available only on Millennium I and Millennium II. It is hardware limitation.
- selection between 1:5:5:5 and 5:6:5 16bpp videomode is done by -rgba option of fbset: "fbset -depth 16 -rgba 5,5,5" selects 1:5:5:5, anything else selects 5:6:5 mode.
- text mode uses 6 bit VGA palette instead of 8 bit (one of 262144 colors instead of one of 16M colors). It is due to hardware limitation of Millennium I/II and SVGALib compatibility.

Benchmarks

It is time to redraw whole screen 1000 times in 1024x768, 60Hz. It is time for draw 6144000 characters on screen through /dev/vcsa (for 32bpp it is about 3GB of data (exactly 3000 MB); for 8x16 font in 16 seconds, i.e. 187 MBps). Times were obtained from one older version of driver, now they are about 3% faster, it is kernel-space only time on P-II/350 MHz, Millennium I in 33 MHz PCI slot, G200 in AGP 2x slot. I did not test vgacon:

```
NOACCEL
      8x16                12x22
      Millennium I  G200   Millennium I  G200
8bpp    16.42         9.54   12.33         9.13
16bpp   21.00        15.70   19.11        15.02
24bpp   36.66        36.66   35.00        35.00
32bpp   35.00        30.00   33.85        28.66

ACCEL, nofastfont
      8x16                12x22               6x11
      Millennium I  G200   Millennium I  G200   Millennium I  G200
8bpp     7.79         7.24   13.55         7.78   30.00        21.01
16bpp    9.13         7.78   16.16         7.78   30.00        21.01
24bpp   14.17        10.72   18.69        10.24   34.99        21.01
32bpp   16.15      16.16   18.73        13.09   34.99        21.01

ACCEL, fastfont
      8x16                12x22               6x11
```

```
      Millennium I  G200   Millennium I  G200   Millennium I  G200
8bpp     8.41         6.01    6.54         4.37   16.00        10.51
16bpp    9.54         9.12    8.76         6.17   17.52        14.01
24bpp   15.00        12.36   11.67        10.00   22.01        18.32
32bpp   16.18        18.29*  12.71        12.74   24.44        21.00

TEXT
      8x16
      Millennium I  G200
TEXT     3.29         1.50

* Yes, it is slower than Millennium I.
```

Dualhead G400

Driver supports dualhead G400 with some limitations:

- secondary head shares videomemory with primary head. It is not problem if you have 32MB of videoram, but if you have only 16MB, you may have to think twice before choosing videomode (for example twice 1880x1440x32bpp is not possible).
- due to hardware limitation, secondary head can use only 16 and 32bpp videomodes.
- secondary head is not accelerated. There were bad problems with accelerated XFree when secondary head used to use acceleration.
- secondary head always powerups in 640x480@60-32 videomode. You have to use fbset to change this mode.
- secondary head always powerups in monitor mode. You have to use fbmatroxset to change it to TV mode. Also, you must select at least 525 lines for NTSC output and 625 lines for PAL output.
- kernel is not fully multihead ready. So some things are impossible to do.
- if you compiled it as module, you must insert i2c-matroxfb, matroxfb_maven and matroxfb_crtc2 into kernel.

Dualhead G450

Driver supports dualhead G450 with some limitations:

- secondary head shares videomemory with primary head. It is not problem if you have 32MB of videoram, but if you have only 16MB, you may have to think twice before choosing videomode.
- due to hardware limitation, secondary head can use only 16 and 32bpp videomodes.
- secondary head is not accelerated.
- secondary head always powerups in 640x480@60-32 videomode. You have to use fbset to change this mode.
- TV output is not supported
- kernel is not fully multihead ready, so some things are impossible to do.
- if you compiled it as module, you must insert matroxfb_g450 and matroxfb_crtc2 into kernel.

Petr Vandrovec <vandrove@vc.cvut.cz>