# XF86Config Demystified

### Contents



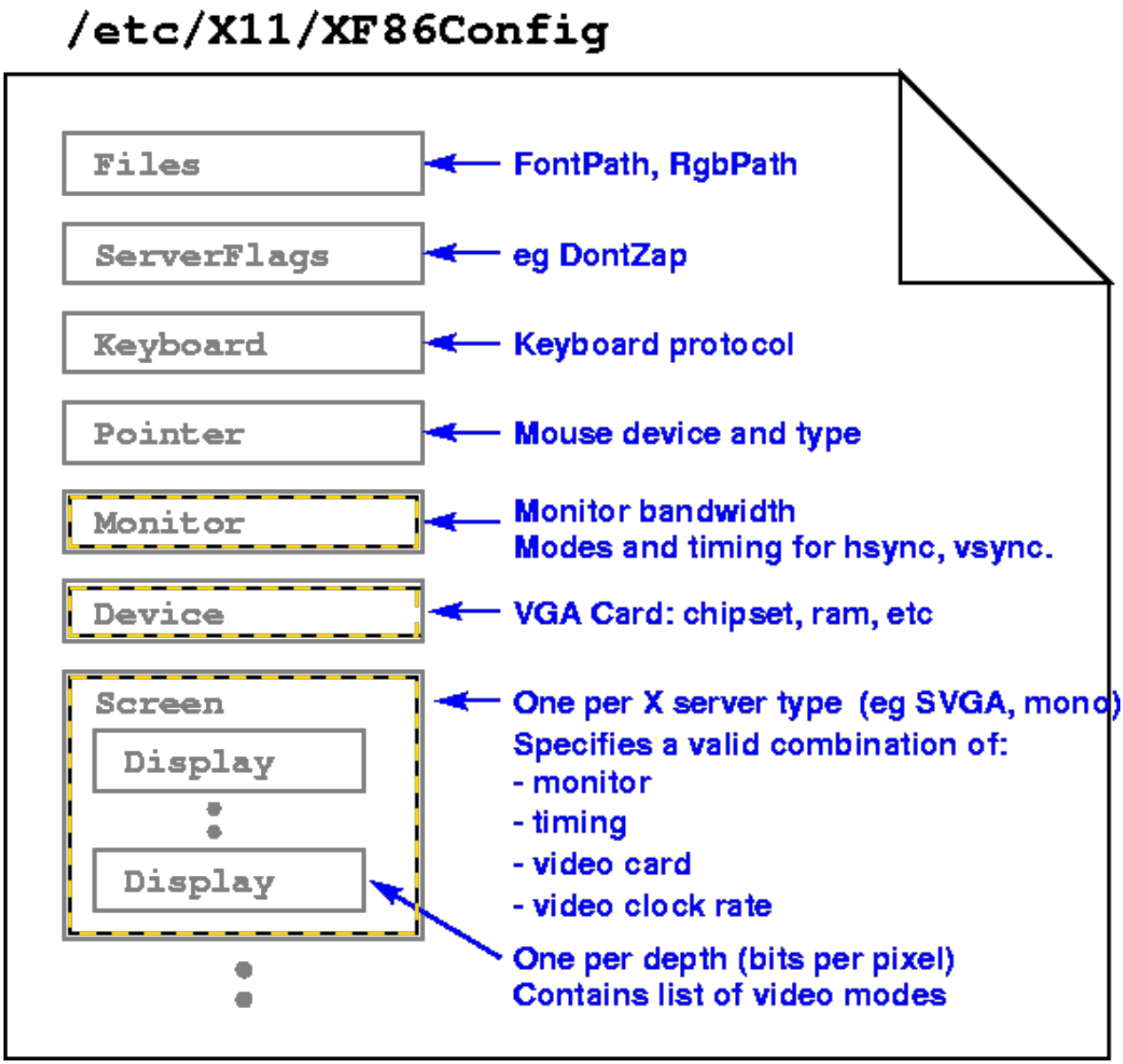
## Introduction

Often, the most difficult part of installing Linux (the first time) is getting X11 working. Programs like `Xconfigurator` (from Red Hat) and `XF86Setup` (from XFree86) go a long way to simplifying the process, but when things go wrong, the simplest solution is to edit the `XF86Config` file manually.

The `XF86Config` file, is divided into sections, each section controlling a specific aspect of the X11 "server". The sections are not all independant, and understanding how they interact is an essential step towards successfully configuring XFree86.

In general, it is the sections highlighted in black/yellow which give the most trouble.

## Anatomy of the xf86config file




**XF86Config sections**

| Section | Description | Important Entries |
|---|---|---|

| | | |
|---|---|---|
| Files | Location of fonts and RGB colorname file | Default is OK. |
| ServerFlags | Things like "`DontZap`" (disables Ctrl-Alt-BackSpace) and "`AllowMouseOpenFail`". | Not normally required. |
| Keyboard | Keyboard protocol and other characteristics. | Default is OK. |
| Pointer | Mouse device and mouse type. | `Device, Protocol, Emulate3Buttons` |
| Monitor | Monitor characteristics ie bandwidth. These *cannot* be probed, so you *must* specify them correctly.<br><br>Also contains `Modeline` entries. Each `Modeline` entry describes video setting, ie. horizontal and vertical resolution, and other timing parameters of the video signal. | `HorizSync, VertRefresh, Modeline` |
| Device | Video card characteristics. Usually probed successfully.<br>If your video card needs special settings (as described in the release notes) this is where to put them. | `VideoRam, Chipset, Ramdac, ClockChip` |
| Screen | Each "Screen" section defines a valid combination of:<br><br>• X-server (SVGA, MONO, VGA16 or "accel")<br>• monitor<br>• video card | `Driver, DefaultColorDepth` |
| Display | One or more "Display" sections may appear within each "Screen" section.<br><br>Each "Display" section contains the list of desired resolutions ("modes") for a particular display-depth (ie. bits per pixel). | `Depth, Modes` |

# Interaction between sections

Although the different sections may appear "independant", they are not. It is important to understand exactly how the sections are related.

The sections `Monitor, Device, Screen` and `Display` all play a role in the selection of a video-mode. This is what happens when the X-server is started:

1. The appropriate `Screen` section is chosen.

   The appropriate `Screen` section is the one with the right `Driver` entry for the X-server program being run.

   The right `Driver` entry the different X-programs is as follows:

| | |
|---|---|
| `XF86_S3 XF86_S3V XF86_W32 XF86_Mach* etc` | `Driver "accel"` |
| `XF86_SVGA` | `Driver "svga"` |
| `XF86_VGA16` | `Driver "vga16"` |
| `XF86_MONO` | `Driver "mono"` |
| `XF86_FBdev` | `Driver "fbdev"` |

2. The `Monitor` and `Device` entries are used to select the corresponding `Monitor` and `Device` (video card) sections, based on the `Identifier` entry in those sections.

   The `Identifier` is completely arbitrary. It is simply used as a pointer to the right section.

3. The `Display` subsection is chosen based on the depth (bits per pixel)

   The depth is determined by the entry `DefaultColorDepth`, or by the command-line argument `-bpp`

*This is where things get tricky....*

4. The `Modes` entry defines a list of *modenames*.

   There must be one or more `Modeline` entries corresponding to each modename. The `Modeline` entries appear in the `Monitor` section.

The first modename in the list is the one which will be used when the X-server starts. The other modes can be accessed using the 3-key combinations:

```
<Ctrl> <Alt> <Keypad_Plus>
```

and

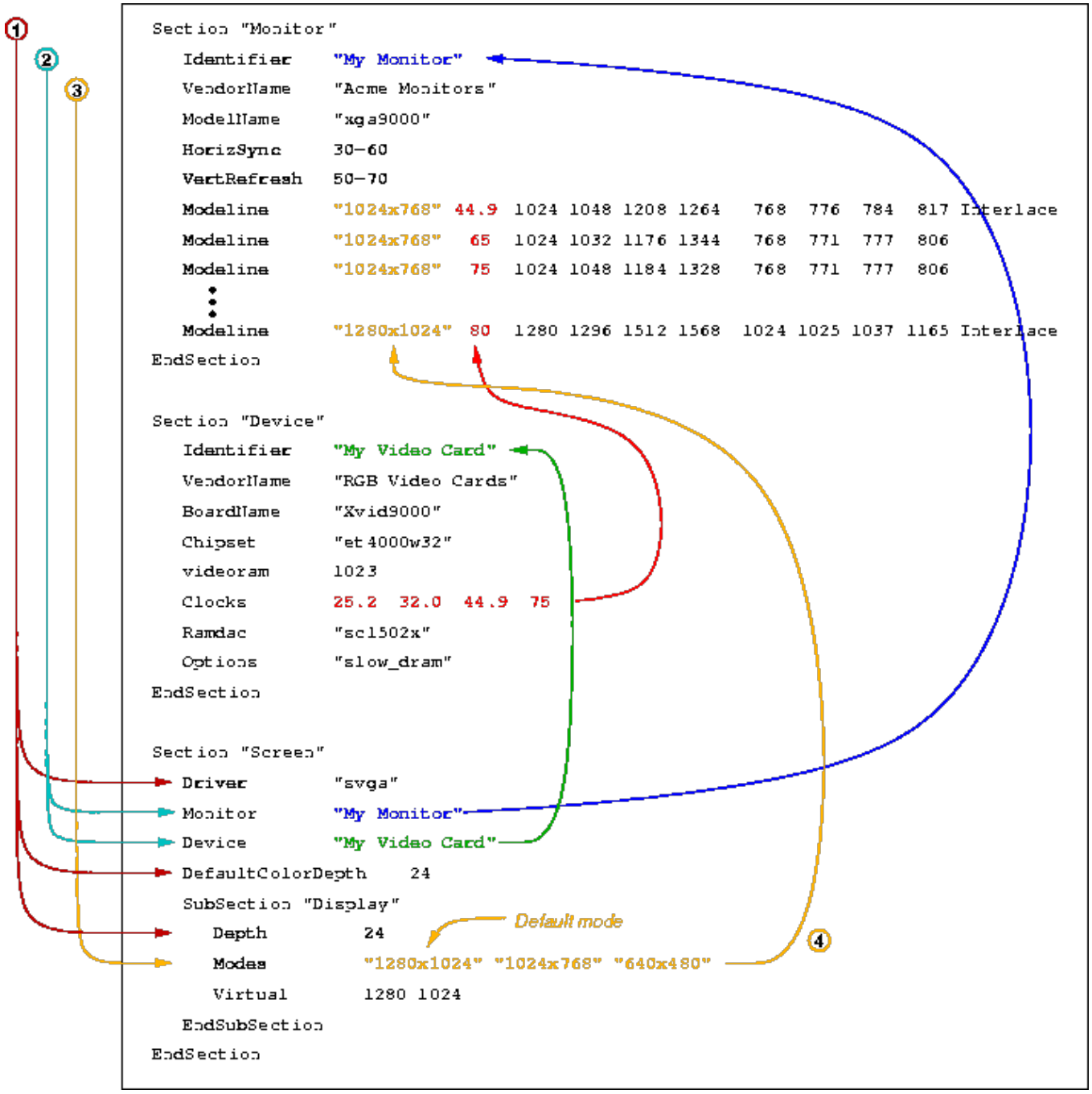
```
<Ctrl> <Alt> <Keypad_Minus>
```

5. Where there is more than one `Modeline` entry with the same name, the following criteria is used to select the mode:
   a. The first numeric parameter of each `Modeline` is the video clock rate. If there is a `Clocks` entry in the `Device` section, then the choice of possible modes is restricted to those `Modelines` with a clock-rate equal to one of those listed in the `Clocks` entry.
   b. The mode found which has the *highest* clock rate, while still within the horizontal and vertical sync range of the monitor (as specified in the `Monitor` section), will be used.

   Most video cards these days do not need a `Clocks` entry. If omitted, any clock rate is considered acceptable, up to the maximum clock rate the chipset supports.

   Note that the modes are referenced by *name*
   eg. "1024x768" is a simple label. It is the timing-values which determine the actual video mode, regardless of the label.

The following diagram illustrates this.




It is generally not necessary to specify the `videoram` or `ramdac`. These parameters can be probed-for.

ISA video cards *require* the `chipset` entry to be present and correct. PCI video cards generally do not require the `chipset` entry (the chipset can be probed by the X-server)

## How to select ...

# How to select the X-server

As mentioned above, the choice of X-server determines the "driver". (Mono, vga16, svga, etc). This means executing the right binary for the X-server. (XF86_SVGA, XF86_VGA16)

### Red Hat

This is set using the symbolic link

```
/etc/X11/X
```

which should link to the corrrect X-server, eg:

```
/etc/X11/X -> ../../usr/X11R6/bin/XF86_SVGA
```

### Debian

This is normally set within the configuration file:

```
/etc/X11/Xserver
```

which looks like this:

```
/usr/bin/X11/XF86_SVGA
Console

The first line in this file is the full pathname of the de
The second line shows who is allowed to run the X server:
RootOnly
Console      (anyone whose controlling tty is on the conso
Anybody
```

Only the first two lines are significant.

# How to select 8, 16, 24, 32 bpp (display depth)

Having more bits-per-pixel gives you more realistic colors. This can be important for any sort of image processing, or simply to get rid of the side-effects of pseudo-color, such as "layering" and poor color matching. However, the improved color rendering of more bits-per-pixel comes at the expense of:

- interactive performance
  ...which can deteriorate significantly
- maximum dot-clock rate
  ...ie refresh rate of the screen in Hz
- maximum resolution
  ...which is limited by the amount of video RAM
  1M byte = 1152x900 @ 8 bpp

There are two ways to set the number of bits-per-pixel:

- from the command-line, using `-bpp` or
- from the XF86Config file, using `DefaultColorDepth`.

## Setting the display depth using -bpp

The X-server program (eg XF86_SVGA) accpets a command-line option:

`-bpp` *`depth`*

However, the server is usually started indirectly, either from the script `startx` or by `xdm` (or `gdm` or `kdm`)

### Setting -bpp using startx

If you are using startx, you can:

- specify -bpp interactively, eg

```
startx -- -bpp 24
```

- Create/modify one of the following files:

  `/etc/X11/xinit/xserverrc` (Red Hat)
  `/usr/X11R6/lib/X11/xinit/xserverrc` (Debian)
  `~/.xserverrc`

  Include the line:

  ```
  X -bpp 24 $@
  ```

### Setting -bpp using xdm

If you are using xdm, modify the file:

`/etc/X11/xdm/Xservers`

To look like this:

```
:0 local /usr/X11R6/bin/X -bpp 24
```

## Setting the display depth using DefaultColorDepth

The other way to set the display depth is to modify the file:

`/etc/X11/XF86Config`

In the appropriate Screen section, add a `DefaultColorDepth` entry of the form.

`DefaultColorDepth` *`depth`*

For example:

```
DefaultColorDepth  24
```

This applies to both xdm and startx. The command-line parameter `-bpp` will override the `DefaultColorDepth` entry, so you can still experiment with

different depths using startx, as shown above.

# How to select the Resolution

The horizontal and vertical resolution is determined by the `Modes` line in the applicable Display section.

Each mode listed on the `Modes` line is a pointer to a corresponding `Modeline` entry in the applicable `Monitor` section. Like the `Monitor` and `Device` names, the mode names are arbitrary eg. the mode name "640x480" could be renamed to "lowres", provided both the `Modeline` entry and the `Modes` line are changed.

For a given list of modes, note that:

a. The first mode in the `Modes` line is the mode which is used when the X-server first starts.
b. the highest resolution becomes the virtual resolution if there is no `Virtual` entry in the `Display` section.
c. Any modes which exceed the `HorizSync` or `VertRefresh` values in the `Monitor` section are deleted.

You can define any number of different modes. The different modes can be reached using the `<Ctrl> <Alt> <Keypad_Plus>` and `<Ctrl> <Alt> <Keypad_Minus>` mechanism.

## Setting the default mode

By default, XF86Setup generates a `Modes` line like this:

```
        Modes     "640x480" "800x600" "1024x768" "1280x1024"
```

Since "640x480" is the first mode in the list, the X-server will always start in that mode. You will need to cycle to the mode you want using `<Ctrl><Alt><Keypad_Plus>` and `<Ctrl><Alt><Keypad_Minus>`

Starting up in "640x480" is quite deliberate, since "640x480" is the most likely to work. This saves you the distress of wondering "whats gone wrong", which would happen if you had started in a higher-resolution mode, and your monitor couldn't handle it.

However, once you're confident all of the listed modes work, you can edit this line, and put your preferred mode first. eg

```
    Modes     "1280x1024" "1024x768" "800x600" "640x480"
```

## Setting the Monitor hsync and vsync range

The last point (c) is very important, because it is the one which is easiest to get wrong, and easiest to fix. For example, the following values will probably not work, or at best, allow only 1 resolution.

```
Section "Monitor"
    ...
    HorizSync 31.5
    VertRefresh 60
    ...
EndSection
```

The reason is in the mathematics:

- A HorizSync of 31.5 kHz allows 31.7 microseconds per horizontal scan line
- A VertRefresh of 60 Hz allows 16667 microseconds per vertical frame

- This *requires* exactly 525 lines per frame

In short, by defining very rigid timing values for our monitor, we have limited our possibilities to a very specific mode. The above values would cause most video modes to be deleted, because they are "beyond the capabilities of the monitor" (as specified). If we're lucky, we might get a 640x480 mode which matches these values.

What we should have specified is something like this:

```
Section "Monitor"
    ...
    HorizSync 30-55
    VertRefresh 50-75
    ...
EndSection
```

These values, taken at opposite extremes allow anything from 400 to 1100 scan lines. The *best* way to set these values is to open the manual for your monitor, and look up the specifications. These will give you the right values to put in here.

Don't go wild and enter really extreme values, because you'll end up with video modes that don't work. Or (rumor has it) you may damage the monitor. If you're working without a manual, start with some conservative values, and expand them gradually. The values shown above are fairly conservative for today's monitors.

The symptoms of exceeding the hsync and vsync range of your monitor are typically:

- a blank screen (most common)
- loss of horizontal sync (diagonal lines)
- loss of vertical sync (rolling display)

In the majority of cases, there are enough `Modeline` entries added by default that **setting the correct `HorizSync` and `VertRefresh` range is all that is required** to get very good results.

## Tweaking the Modeline entries

Modeline entries define three things, from the users perspective:

- horizontal and vertical resolution
- refresh rate
- location of the image on the screen (height/width/position)

These aspects are limited by:

- amount of video RAM

  1 M allows up to 1152x900 @ 8 bpp
  4 M allows up to 1280x1024 @ 24 bpp

- Monitor hsync and vsync range

  In general, higher resolutions require higher frequencies, especially for the HorizSync range.

- Maximum "dot-clock" of the video card

  ie. the maximum rate at which the video card can send pixels to the monitor

- RAMDAC Bandwidth

  Each video card has a device known as a RAMDAC - a very fast digital to analogue converter, which converts the binary video data into three analogue video signals (red green and blue).

- Monitor Bandwidth

ie. the maximum rate at which the monitor can receive pixels from the video card.
Higher resolutions require higher monitor bandwidth (no exceptions :-)
The first symptom of exceeding the monitor bandwidth is dull colors, and a color balance which changes with the colors on the screen.

XFree86 is designed to choose video modes which meet all of the above constraints.

### Using X -probeonly

If you're not getting the modes you want, or think you should, the first thing you should do is run:

```
X -probeonly >& /tmp/X.log
```

Let's examine the output in detail, using my S3 Virge PCI card as an example.

Each line is prefixed with either `(**)` or `(--)`

`(**)`
: means that this information was obtained from the `XF86Config` file

`(--)`
: means that this information was probed for or calculated by the X-server

Note what the following lines are telling me about my configuration:

```
(--) SVGA: Mode "1024x768" needs vert refresh rate of 86.96 Hz.
Deleted.
(--) SVGA: Mode "1152x864" needs hsync freq of 70.88 kHz.
```

`Deleted.`

This is telling me that these modes are beyond the capabilities of my monitor. In this case, there are other modes with lower frequencies which give me the same resolutions, so we won't worry about them.

If a particular mode is only slightly outside our range I could either:

- Adjust the range for `HorizSync` or `VertRefresh`
- Adjust the dot-clock value (usually this means decreasing it)

`(--) SVGA: Ramdac speed: 135 MHz`

This sets the practical limit of my clock rate

`(--) SVGA: Maximum allowed dot-clock: 135.000 MHz`

This is the limit of my clock rate, as determined by the chipset and RAMDAC speed.

This means that any `Modeline` with a clock of >135 will be ignored.

```
(**) SVGA: Mode "640x480": mode clock = 31.500
(**) SVGA: Mode "800x600": mode clock = 40.000
(**) SVGA: Mode "1024x768": mode clock = 75.000
(**) SVGA: Mode "1280x1024": mode clock = 110.000
```

These are the modes which the X-server has selected from all those modes with the same name.

`(**) SVGA: Using 8 bpp, Depth 8, Color weight: 666`

My X-server has defaulted to using 8 bpp.

Let's see what I get with 24 bpp.

**`X -probeonly -bpp 24 >& /tmp/X.log`**

```
(--) SVGA: Ramdac speed: 135 MHz (57 MHz for 24 bpp)
(--) SVGA: Maximum allowed dot-clock: 57.000 MHz
```

Going to 24 bpp has slashed my maximum dot-clock from 135 MHz

to 57 MHz. This will affect the two high-resolution modes, "1024x768" and "1280x1024"

```
(**) SVGA: Using 24 bpp, Depth 24, Color weight: 888
```

Just checking :-)

```
(--) SVGA: Clock for mode "1280x1024" is too high for the
configured hardware.
        Limit is 57.000 MHz
(--) SVGA: Removing mode "1280x1024" from list of valid modes.
```

Notice that "1280x1024" has been eliminated completely. This is because there was no `Modeline` entry named "1280x1024" with a clock of 57 MHz or less.

```
(**) SVGA: Mode "1024x768": mode clock = 57.000
```

"1024x768" has been down-graded from 75 Mhz to 57 MHz, which reduces my refresh-rate.

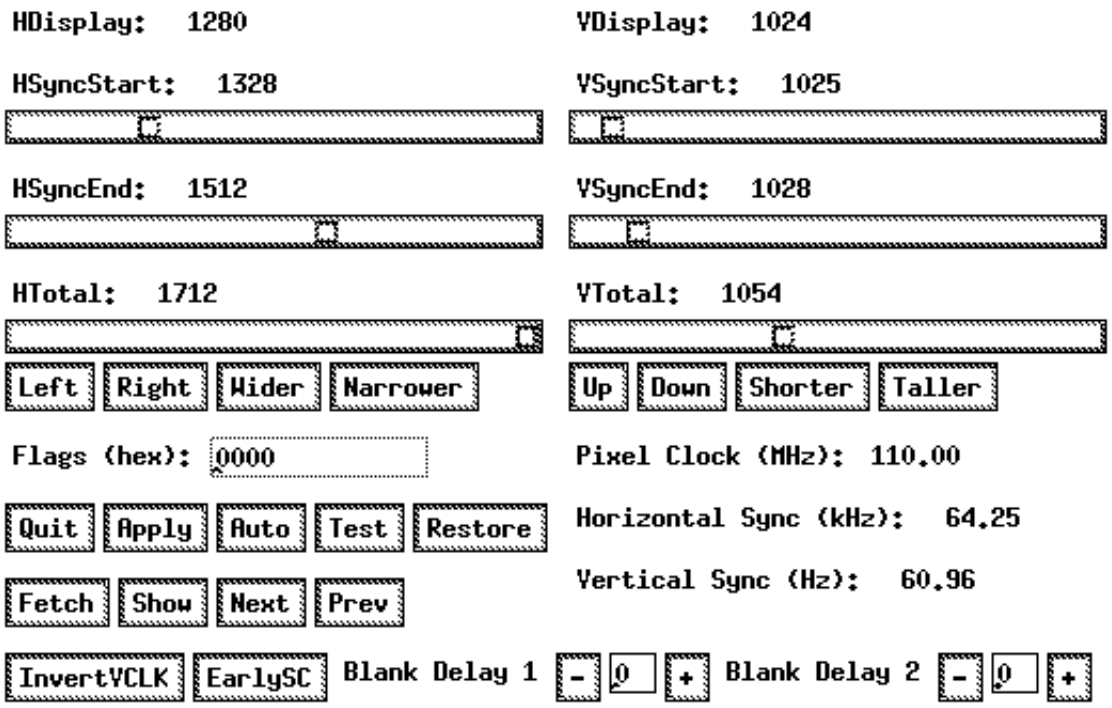
```
(--) SVGA: Virtual resolution set to 1024x768
```

Since I have not specifed a virtual resolution in the `Display` section, the highest resolution is used. In this case, it is 1024x768 because 1280x1024 was deleted.

Remember that some modes are bound to be beyond the capabilities of the hardware, as was 1280x1024 @ 24bpp in this instance

### Using xvidtune

`xvidtune` allows you to center the image on the screen by adjusting with `Modeline` values. This avoids the situation where you have to adjust the monitor settings every time you change modes.




The changes are not visible until you click [Auto] or [Test] or [Apply].

Once you have a setting you like, click on [Show] . The Modeline values are written to the terminal, like this:

```
"1280x1024"   110.00   1280 1404 1588 1764   1024 1025 102
```

These can then be copied and pasted into the `XF86Config` file.

### More about Modelines

If you want more infomation about modelines, and what the numbers actually mean, see the document "XFree86-Video-Timings-HOWTO" (.gz)

Red Hat:
/usr/doc/HOWTO/other-formats/html/XFree86-Video-Timings-HOWTO.html
Debian:
/usr/doc/HOWTO/html/XFree86-Video-Timings-HOWTO.html

# Debugging Exotic Hardware

If you have a very new or unusual video card, or are experiencing problems, the first port of call should be the XFree86 documentation. This is available at:

> http://www.xfree86.org/3.3.3.1/index.html
> http://xfree86.mirror.aarnet.edu.au/3.3.3.1/index.html

The documentation contains a wealth of useful information for specific video hardware, including additional keywords which can be added to the Device section.

In a pinch, adding the following `Option` lines to the `Device` section can often get you a working system:

```
Section "Device"
    ...
    Option "no_accel"
    Option "sw_cursor"
    ...
EndSection
```

However, they *will* severely degrade performance. The XFree86 documentation often provides alternatives which will sacrifice little or no performance.

# How to use two pointing devices (mice)

XFree86 only allows one pointing device to be used at a time. This is inconvinient on many laptop computers, which usually have a built-in mouse-device, but may also have a conventional mouse attached to a

serial port.

The solution is to use gpm. gpm is capable of taking input from two devices, and providing net result on a named-pipe output, `/dev/gpmdata`

Edit the file:

`/etc/rc.d/init.d/gpm` (Red Hat)

or

`/etc/init.d/gpm` (Debian)

Such that gpm is started with the -R option, and the -M option for the second mouse device, eg:

```
gpm -R -m /dev/psaux -t ps2 -M -m /dev/ttyS0 -t logi
```

You will also need to edit the `XF86Config` file. Sepcifically, the Pointer section:

```
Section "Pointer"
    Device   "/dev/gpmdata"
    Protocol "mousesystems"
    Emulate3Buttons     # if you need it
EndSection
```

Note that when sending data to `/dev/gpmdata` the mousesystems protocol is used by gpm, regardless of the original protcol used by the mouse device(s).

Last Modified: $Date: 2000/12/03 00:52:20 $

*George Hansper* *george@hansper.wattle.id.au*