List of XFree86 documentation

TOC

The XFree86 Project, Inc

23 December 1998

1. Available Documentation

1. Available Documentation

- Release Notes for XFree86[tm] 3.3.5
- README for XFree86[tm] 3.3.5
- Copyright
- Quick Start Guide for XFree86
- Mouse Support in XFree86
- The XInput extension in XFree86
- Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04
- README for XFree86 on FreeBSD
- Information for ISC Users
- Information for Linux Users
- README for XFree86 on LynxOS
- README for XFree86 on NetBSD
- README for XFree86 on OpenBSD
- README for XFree86 on OS/2
- Notes on Rebuilding XFree86/OS2 from Scratch
- Information for SCO Users
- Information for Solaris for x86 Users
- Information for SVR4 Users
- Configuring XFree86
- Building XFree86
- XFree86 Video Timings HOWTO
- Readme for the XFree86 LinkKit
- How to add an (S)VGA driver to XFree86
- Information for using/developing external clock setting programs
- Notes on the AGX Server
- Information for Alliance Promotion Chipset Users
- Information for ARK Logic Chipset Users
- ATI boards README
- Information for Chips and Technologies Users
- Information for Cirrus Chipset Users

- Information for Cyrix Chipset Users
- Information for DEC 21030 Users (aka TGA)
- Information for EPSON SPC8110 Chipset Users
- The Linux/m68k Frame Buffer Device
- Notes for Mach32 X Server
- Notes for Mach64 X Server
- Information for Matrox Millennium Users
- Information for Number Nine I128 Users
- Information for i740 Users
- Information for NeoMagic Chipset Users
- Information for NVidia / SGS-Thomson NV1, Riva 128 and Riva TNT Users
- Information for Oak Technologies Inc. Chipset Users
- XFree86 P9000 Server Release Notes
- Information for Rendition Chipset Users
- Information for S3 Chipset Users
- Information for S3 ViRGE and ViRGE/VX Users
- Information for SiS Users
- Information for 3DLabs Chipset Users
- Information for Trident Chipset Users
- Information for Tseng Chipset Users
- README.VIDEO7
- Information for ET4000/W32 and ET6000 Chipset Users
- Information for Western Digital Chipset Users

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/DocIndex.sgml,v 3.24.2.18
1999/05/29 09:39:05 dawes Exp $
```

```
$XConsortium: DocIndex.sgml /main/15 1996/10/28 05:12:53 kaleb $
```

Release Notes for XFree86[tm] 3.3.5

The XFree86 Project, Inc

17 August 1999

This document describes the bugs fixed and the features added in XFree86 3.3.5 compared with the 3.3.4 release, It also includes installation instructions for the binary distributions. It is strongly recommended that anyone using XFree86 update to version 3.3.5.

---

1. XFree86 and X11R6.3

2. X11R6.4

3. OS issues

4. What's new in 3.3.5?

5. What's new in 3.3.4?

6. What's new in 3.3.3.1?

7. What's new in 3.3.3?

    7.1. Security fixes

    7.2. Bug fixes

    7.3. New Features

    7.4. Known Problems

8. XFree86 and Open Source Software

9. Installing the XFree86 3.3.5 Release

1. XFree86 and X11R6.3

XFree86 releases starting with the 3.2A beta are based on the X Consortium's X11R6.3 (the final release from the X Consortium).

R6.3 is an update to R6.1, and is intended to be compatible with R6.1 and R6 at the source and protocol levels. Binaries should be upward-compatible. X11R6.3 includes some new Xserver extensions: SECURITY, XC-APPGROUP, XpExtension (print extension), and an updated, and standardised version of LBX. X11R6.3 also has new standards, including RX (X Remote Execution MIME type), and a proxy management protocol. X11R6.3 includes support for gzipped fonts.

R6.1 is an update to R6, and is intended to be compatible with R6 at the source and protocol levels. Binaries should be upward-compatible. X11R6.1 includes some new Xserver extensions: DOUBLE-BUFFER, XKEYBOARD and RECORD.

What about R6.2? X11R6.2 is the name given to a subset of X11R6.3, which has only the print extension and the Xlib implementation of vertical writing and user-defined character support in addition to those features included in R6.1.

2. X11R6.4

In September 1998 The Open Group changed the copyright of X11R6.4 from the non-free copyright used for the initial release of X11R6.4 back to the old free X Consortium style copyright. Given the fact that our main development focus is on XFree86-4.0 and that the 3.3.3 release was intended mostly to fix some bugs and get out new drivers to the public, we opted not to include X11R6.4 in XFree86-3.3.3. Since XFree86-3.3.5, XFree86-3.3.4 and XFree86-3.3.3.1 are merely quick bug fix releases to XFree86-3.3.3 (that happen to add support for some new common hardware as well), the same applies here.

XFree86-4.0 will be X11R6.4 based.

3. OS issues

Always check the OS specific README files for special requirements or caveats.

Users running Linux should note that Elf is now the only binary type supported for Linux OSs. This means that binaries for ix86/a.out and AXP/ECOFF are not available with this release.

Users running FreeBSD 3.0 should note that only ELF binaries are provided with this release. a.out shared libraries are also included for compatibility purposes.

4. What's new in 3.3.5?

- Support for S3 Savage4 and Savage3D. Limited to Linux/x86 at this point. Please see README.S3V.
- Support for S3 Trio3D/2X.
- Support for DGux.
- Support for QNX.
- Fix bug in Mach64 server on Rage LT and Rage LT Pro.
- Fix SiS driver for 530 and 620.
- Fix the spurious underline problem on NVidia Riva TNT cards.
- Fix the PS/2 mouse problem with later Linux kernels.
- Misc updates and bugfixes in Rendition driver.
- Updates from SuSE and Red Hat, including more keyboards, PAM support, ARM and AXP fixes, security fixes.

5. What's new in 3.3.4?

- Several security fixes.
- Intel i740 support (donated by Precision Insight).
- SiS 530 and SiS 620 support.
- 3Dfx Voodoo Banshee and Voodoo3 support.
- Trident Blade3D, CyberBlade and Cyber9525 support.
- S3 Trio3D support.
- Matrox G400 support.
- NVIDIA Riva TNT2 support and better acceleration for all Riva chipsets (donated by NVIDIA).
- Rewritten Cyrix MediaGX support (donated by Cyrix). **Warning: this is reported to hang some machines!** If that happens, please use the SVGA server in XFree86-3.3.3.1 instead.
- Acceleration for XF68_FBDev on PPC.
- VMWare's DGA-1.1 extension. Note that the next major release of XFree86 will NOT include DGA-1.1 but the newly developed DGA-2.0 that contains significantly more features than DGA-1.1 and will most likely not be compatible with DGA-1.1
- Change xterm to use the tty default value for the backspace key.
- Japanese documentation and manpage updates.
- Updates and new hardware support (Acecad flair, Calcomp DrawingBoard) for xinput extension.
- Bug fixed for cards with S3 Aurora64V+ (M65) chip, VGA output should now work.

6. What's new in 3.3.3.1?

- A system clock slowdown caused by 3Dlabs driver has been fixed.
- Drawing bugs with C&T HiQV chips have been fixed.
- Drawing problems in the Cyrix driver have been fixed.
- The Matrox G100/G200 PCI versions should now be fully supported.
- The Mach64 server now supports gamma correction.
- Open Source NVIDIA driver has been provided.
- I128 Rev IV support has been added.
- Another S3V lockup has been fixed.
- A drawing bug in cfb24 has been fixed.
- A problem causing lockups with some Trident cards has been fixed.
- Updates for SCO, FreeBSD, Linux glibc OS support.
- DG/ux support has been added.
- GNU/Hurd support has been added.
- Several XINPUT problems have been addressed.
- DGA relative mouse movement events when XINPUT is defined have been fixed, as have DGA-related problems with the NVIDIA and S3V drivers.
- The X server now reads Xauthority files using the real user id.
- Several small fixes to core clients.
- A bug in Xlib's handling of KOI8-R has been fixed.
- PC98 cards database, sample config file and XKB handling have been fixed.

7. What's new in 3.3.3?

7.1. Security fixes

- Several buffer overrun problem discovered since the release of XFree86-3.3.2 have been fixed
- Sanity checks on DISPLAY variable
- Attempt to stop X connection hijacking (sticky bit for /tmp/.X11-unix) Note that this is only a short-term partial solution, and it is doesn't help at all for some SYSV based OSs (like Solaris 2.x).

7.2. Bug fixes

- Fix a serious LBX bug using uninitialized variables.
- Fix some Xlib bugs that cause problems when using XKB in some locales (like latin2), add support for iso8859-15, and include a couple of basic fonts for iso8859-15
- Fix xf86config to handle more than 10 modes and to be prepared for XFCom / XBF servers.
- Lots of xterm changes, see xterm.log.html in the sources.
- Fix problems with high dot clocks in high color depths on Riva128.
- Fix problem in the S3 drivers with disabled onboard S3 chips when using S3 cards.
- Fix problems with Cirrus 5480 at high resolutions and jitter that appeared with the 546x's using the BitBLT engine.
- Fix clock limits in some cases in Tseng driver.
- Fix some lockups with ViRGE chips.
- Improved timing calculations for video FIFO in the Mach64 X server.
- Fixed bug in font rendering code in the Mach64 X server.
- Fixed VGA font restoration bug when exiting the Mach64 X server.
- Several XF68_FBDev fixes.
- Fix wrong clock limits for S3 Trio64V+.
- Fix some generic rendering errors in cfb and vga code.
- Fix text restore problems and improve high res 32bpp modes in MGA driver; fix 24bpp and 32bpp display problems; disable probing for memory on some MGA chipsets; fix maximum blit size; fix sync on green for Mystique.
- Fix problems with Xnest crashing with too many visuals.
- Fixes for 64bit architectures.
- Fix cursor bug in S3V server.
- Fixes for memory probing, max dotclock probing and DPMS display off on C&T chipsets.

- Fix LCD detection for CLGD755x and the double mouse bug and the blanking bug in the cirrus driver.
- Fix some problems with -quiet flag (where some variables stayed uninitialized)
- SuperProbe can detect C&T HiQV chips now, with an exception in the case of No-PCI bus connected. The "-no_bios" option of SuperProbe solve this situation.
- The C&T chipsets now use software cursors by default to avoid a number of minor problems in certain circumstances. Hardware cursors can still be used by adding the "hw_cursor" option to XF86Config.
- EGC server now works on Linux/98. XF98Setup also works with it.
- Fix VT switch problem with MGA server on Linux/98.

7.3. New Features

- New driver for Cyrix MediaGX based motherboards.
- New driver for Rendition V1000 and V2x00 chipsets (not accelerated).
- New driver for Weitek P9100 based cards.
- New driver for SiS 5597/98 and SiS 6326; treat SiS 6215 and 6225 as 6205.
- New server for 3Dlabs based cards using GLINT 500TX and MX (with IBM RAMDAC), Permedia (with IBM RAMDAC), Permedia 2 and Permedia 2v.
- Support for the Matrox G100 and G200 based cards to the MGA driver.
- Support for C&T 69000 and 32bpp on 65550 and later.
- Support for NeoMagic notebook chipsets.
- Support for EPSON SPC8110.
- Support for NVidia Riva TNT.
- Acceleration for Trident Image975, Image985, Cyber9397, Cyber9388.
- Support for the new ATI Rage Pro, VT4 and Rage IIC based cards has been added.
- 24-plane TGA support.
- Config support to build XFree86 on Linux with DECnet transport.
- Support to build XFree86 for FreeBSD/ELF.
- Support for vesafb on Linux/x86.
- LynxOS 3.0 support.
- Updates to SuperProbe.
- New XInput drivers for AceCad ADVANCEDigitizer, MicroTouch TouchPen, SGI dial box.
- Add local font directory.

7.4. Known Problems

The problems listed here are those known at the time of the release. See the XFree86 FAQ for more up to date information.

- There are problems with some Cirrus laptop chipsets (75xx). The driver seems to work for some people, but not others. Until someone with the appropriate hardware can look into this, these problems are unlikely to be fixed. If you wish to work on this, please contact us. We don't need testers, we need people willing and able to fix the problems.
- There are problems with some of the Trident laptop chipsets. The driver seems to work in a limited way for some people, but not others. Until someone with the appropriate hardware can look into this, these problems are unlikely to be fixed. If you wish to work on this, please contact us. We don't need testers, we need people willing and able to fix the problems.
- SuperProbe command fails to detect some newly supported chips. Currently, the probing result with Xserver itself with appropriate setting in XF86Config (and maybe option "-probeonly") can be more relied on than the result from SuperProbe for newer chips. If you are interested in improving the design of SuperProbe's code, let's come and join as the member of the XFree86 ``developer team".

Release Notes for XFree86[tm] 3.3.5 **:** *What's new in 3.3.3?*
Previous: *What's new in 3.3.3.1?*
Next: *XFree86 and Open Source Software*

8. XFree86 and Open Source Software

XFree86 public releases in general follow the Open Source Software definition as set forth at http://www.opensource.org/osd.html. This definition is actually a subset of our requirements.

All code in XFree86 3.3.5 satisfies the Open Source Software definition.

9. Installing the XFree86 3.3.5 Release

The XFree86 3.3.5 binaries are distributed as a full release.

NOTE: the X servers are no longer installed setuid root. If you are starting your X servers with startx/xinit, or something similar, you will need a copy of the setuid Xwrapper, and an updated xinit. These can be found in Xbin.tgz.

What follows is a list of the XFree86 3.3.3 components. There may be some variations in this for some OSs.

The following are required for all new installations, or when upgrading from a version older than 3.3:

```
preinst.sh      Pre-installation script
postinst.sh     Post-installation script
extract         XFree86 extraction utility
Xbin.tgz        Clients, run-time libs, and app-defaults files
Xdoc.tgz        Documentation
Xfnts.tgz       75dpi, misc and PEX fonts
Xlib.tgz        Data files required at run-time
Xman.tgz        Manual pages
Xset.tgz        XF86Setup utility
Xjset.tgz       XF86Setup utility (if you prefer the Japanese version)
XVG16.tgz       16 colour VGA server (XF86Setup needs this server)
Xcfg.tgz        sample config files for xinit, xdm
```

The following are required when upgrading from version 3.3 or later:

```
preinst.sh      Pre-installation script
postinst.sh     Post-installation script
extract         XFree86 extraction utility
Xbin.tgz        Clients, run-time libs, and app-defaults files
Xdoc.tgz        Documentation
Xlib.tgz        Data files required at run-time
Xman.tgz        Manual pages
Xset.tgz        XF86Setup utility
Xjset.tgz       XF86Setup utility (if you prefer the Japanese version)
XVG16.tgz       16 colour VGA server (XF86Setup needs this server)
```

While it isn't essential to update the standard fonts, this version does include some minor fixes to some of them, as well as the addition of two basic ISO 8859-15 fonts. If you want to upgrade the standard fonts you will also need:

```
Xfnts.tgz       75dpi, misc and PEX fonts
```

NOTE: Be very careful about installing Xcfg.tgz over an existing installation if you have customised your xinit and/or xdm config files. Installing Xcfg.tgz will overwrite any existing files. If you do have customised files, there is no need to install Xcfg.tgz.

NOTE: The bitmap fonts distributed with this release are compressed using gzip rather than compress. This means that you will probably want to remove the old versions (after backing them up). The Xservers and font server in releases prior to 3.2A cannot read gzipped fonts, so keep a copy of the old fonts if you wish to run older servers.

The following X servers are for PC/AT based hardware (i.e., typical Intel ix86 based PCs). Choose at least one which matches your hardware, as well as the VGA16 server. The VGA16 server is required by the new configuration utility (XF86Setup). A

list showing which X server is required for a range of video cards can be found at http://www.xfree86.org/cardlist.html.

```
X3DL.tgz        3Dlabs server
X8514.tgz       8514/A server
XAGX.tgz        AGX server
XI128.tgz       I128 server
XMa32.tgz       Mach 32 server
XMa64.tgz       Mach 64 server
XMa8.tgz        Mach 8 server
XMono.tgz       Mono server
XP9K.tgz        P9000 server
XS3.tgz         S3 server
XS3V.tgz        old S3 ViRGE server (please use SVGA server)
XSVGA.tgz       SVGA server
XVG16.tgz       16 colour VGA server (XF86Setup needs this server)
XW32.tgz        ET4000/W32, ET6000 server
```

The following X servers are available for Alpha hardware:

```
XMa64.tgz       Mach 64 server
XMono.tgz       Mono server (generic driver only)
XP9K.tgz        P9000 server
XTGA.tgz        DEC 21030 (TGA) server
XS3.tgz         S3 server
XS3V.tgz        old S3 ViRGE server (please use SVGA server)
XSVGA.tgz       SVGA server (Matrox Millennium and S3 ViRGE drivers only)
```

The following X servers are for PC98 hardware. Note that PC98 is a Japanese computer standard and has nothing to do with Win98, or the Intel and Microsoft PC98 specification. If you have a PC98 machine, choose one which suits your hardware. If you don't know what a PC98 machine is, you don't need any of these. These servers **will not run** on "normal" PCs, so don't even try them if you don't have a Japanese PC98 machine. A list showing which X server is required for a range of PC98 video cards and computers can be found at http://www.xfree86.org/cardlist98.html.

```
X9NS3.tgz       PC98 NEC(S3) server
X9SPW.tgz       PC98 PCSKB-PowerWindow(S3) server
X9LPW.tgz       PC98 PowerWindowLB(S3) server
X9EGC.tgz       PC98 EGC(generic) server
X9GA9.tgz       PC98 GA-968V4/PCI(S3 968) server
X9GAN.tgz       PC98 GANB-WAP(cirrus) server
X9480.tgz       PC98 PEGC-480(generic) server
X9NKV.tgz       PC98 NKV-NEC(cirrus) server
X9WS.tgz        PC98 WABS(cirrus) server
X9WEP.tgz       PC98 WAB-EP(cirrus) server
X9WSN.tgz       PC98 WSN-A2F(cirrus) server
X9TGU.tgz       PC98 TGUI server
X9MGA.tgz       PC98 MGA server
X9SVG.tgz       PC98 CLGD755x server
X9set.tgz       PC98 XF98Setup utility
```

The following are optional.

```
Xf100.tgz       100dpi fonts
Xfcyr.tgz       Cyrillic fonts
Xfnon.tgz       Other fonts (Chinese, Japanese, Korean, Hebrew)
Xfscl.tgz       Scalable fonts (Speedo and Type1)
Xfsrv.tgz       Font server and config files
```


```
Xprog.tgz       X header files, config files and compile-time libs
Xnest.tgz       Nested X server
Xvfb.tgz        Virtual framebuffer X server
Xprt.tgz        X Print server
Xps.tgz         PostScript version of the documentation
Xhtml.tgz       HTML version of the documentation
Xjdoc.tgz       Documentation in Japanese (for version 3.3.4)
Xjhtm.tgz       HTML version of the documentation in Japanese (3.3.4)
Xlkit.tgz       X server LinkKit
Xlk98.tgz       X server LinkKit for PC98 servers
```

If you already have a version of XFree86 installed, **MAKE A BACKUP OF `/usr/X11R6` BEFORE DOING ANYTHING ELSE**. The standard installation procedure will overwrite your existing version of XFree86.

If you are installing from scratch, create a directory called `/usr/X11R6`, then extract the required `.tgz` files. If you don't have enough space in `/usr` for this, create a directory elsewhere and create a symbolic link to it. E.g., if you create a directory in `/home`:

```
mkdir /home/X11R6
ln -s /home/X11R6 /usr
```

The next step is to run the pre-installation script. This script makes some preliminary checks of your system. For some OSs, it may tell you to install new versions of some system components before proceeding with the installation. This script may also remove some outdated files and symbolic links from a previous installation that could cause problems.

For the purposes of these installation instructions, it is assumed that you have downloaded all the files to the `/var/tmp` directory. If you've put them in another directory, that's fine -- just replace all occurrences of ``/var/tmp'' with the name of that directory.

To run the pre-installation script, go to `/usr/X11R6` and run it:

```
cd /usr/X11R6
sh /var/tmp/preinst.sh
```

The next step is to make the installation utility executable. To do this, make sure the `extract' file is in the same directory as all the X*.tgz files, and run the following from that directory:

```
chmod 755 extract
```

The installation utility ``extract'' is used to unpack the .tgz files that make up the XFree86 distribution. The .tgz files are gzipped tar files. However, ``tar'' in its standard form on most OSs is not well-suited to the task of installing XFree86. The extract utility is a modified version of GNU tar 1.12 built with the options required to make it suitable for installing XFree86. The source for extract is available from the same place you got the XFree86 distribution.

It is strongly recommended that you use the provided extract utility to unpack the XFree86 distribution. If you choose to ignore this and use something else, we don't want to hear from you if you run into problems. It is also important that you do not rename the extract utility. If renamed, it behaves just like the normal GNU tar.

To extract the XFree86 binaries, run the following as **root**:

```
cd /usr/X11R6
/var/tmp/extract /var/tmp/X*.tgz
```

Once the required `.tgz` files have been extracted, run the post installation script:

```
cd /usr/X11R6
sh /var/tmp/postinst.sh
```

For OSs which use ldconfig, you may need to run ldconfig or reboot to complete the installation. The postinst.sh script s
run ldconfig correctly for you if you are using Linux, FreeBSD, NetBSD or OpenBSD. For other OSs that use ldconfig, c
how it normally gets run at boot time.

Once the installation is complete, you should run the one of the configuration utilities (`XF86Setup` or `xf86config`) to configure the X server. This is essential for a new installation but optional for an existing installation. Refer to the QuickStart document for configuration information.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/RELNOTE.sgml,v 3.59.2.79 1999/08/17
07:39:30 hohndel Exp $
```

Release Notes for XFree86[tm] 3.3.5 **:** *Installing the XFree86 3.3.5 Release*
Previous: *XFree86 and Open Source Software*
Next: *Release Notes for XFree86[tm] 3.3.5*

Quick-Start Guide to XFree86 Setup

TOC

Joe Moss

27 February 1998

Current releases of XFree86 include several tools that can help to automate the process of server configuration. Much of the existing documentation, however, describes how to do the job manually, including many technical details. For those users with esoteric hardware or with the desire to get their hands dirty under the hood, this is great, but many users are using common hardware and just want to get X up and running quickly. This guide is for them.

---

1. Before You Start

2. What to Do - An Overview

3. Using `XF86Setup`

3.1. Initial questions
3.2. Configuration areas
3.3. Back to text mode
3.4. The second server
3.5. Ending text

4. Running `xf86config`

4.1. The intro screen
4.2. Getting your `PATH` right
4.3. Mouse setup
4.4. Keyboard setup
4.5. Monitor setup
4.6. Selecting your card
4.7. Server selection
4.8. Screen/Video configuration
4.9. Mode Selection
4.10. Creating the `XF86Config` file
4.11. Some final notes

TOC

5. Fixing the XF86Config file

6. Running `xvidtune`

7. Troubleshooting

7.1. The mouse doesn't move correctly, it stays in one area of the screen

7.2. The server doesn't start, it says the mouse is busy.

7.3. The middle button doesn't work.

7.4. The display is shifted to the left/right/top/bottom

7.5. I don't appear to have xf86config or xvidtune on my system

---

1. Before You Start

There are a few bits of information that you will need to have before you can setup the server:

The model name of your video card

Make sure you know the exact model name of the card. It may help to also know the graphics chipset, RAMDAC, and clock chip used on your card.

The amount of memory on your video card

Find out how many megabytes of RAM are on your video card.

Whether or not your card is VGA compatible

Most cards these days are VGA compatible, but for example, if you have and older monochrome card, it might not be.

Your monitor's specifications

Specifically, you need to know the horizontal sync rate(s), and vertical refresh rate(s). These are **important**! Consult your monitor's manual.

The protocol used by your mouse

It will help speed up the process, if you know which protocol is used by your mouse to communicate. Some mice are capable of using two different protocols, although the method of switching between them varies. In some cases, with new Plug-n-Play mice, the protocol can be determined automatically.

2. What to Do - An Overview

There are three tools that can be used to set up XFree86:

- XF86Setup
- xf86config
- xvidtune

XF86Setup primarily uses a graphical user interface and is the preferred tool for initial setup, but there are a few cases where it can't be used. If you are using a card that is not VGA compatible, have a fixed-frequency monitor, or are running OS/2, you'll not be able to use XF86Setup, read about xf86config instead. If you have limited RAM or a slow system, you might be better off using xf86config as well.

The xf86config program is text based only, but works for almost any hardware combination. If you have a fixed frequency monitor that won't work with standard text modes, you will have to read the necessary documentation and do the configuration manually.

To get things looking just right, you may need to use `xvidtune`, a program that allows you to make adjustments to the displayed image (e.g. make it wider, move it a little to the left, etc.). XF86Setup will allow you to run xvidtune at the appropriate time; if you use xf86config, you can use xvidtune afterwards.

All of these are explained in detail in the following sections. If you're the type that doesn't like to read the documentation, but would rather just try and figure your way through things, you can just type `XF86Setup` now. If you have problems, the documentation will still be here.

Although it is possible to use XF86Setup from within X to make changes to your existing configuration, such use is not specifically documented here. These instructions are primarily for those initially setting up XFree86 on their system.

3. Using XF86Setup

XF86Setup will first check around to make sure certain files are installed and that you are running as root. If a problem is found, it will display a message and exit. Correct the problem (e.g. install the missing files) and run it again.

XF86Setup is internationalized. If you are Japanese and set the LANG environment variable to ja, japan, japanese, etc., XF86Setup's screen can be Japanized. But it is necessary that XF86Setup is built with Japanized Tcl/Tk. Other language can be added, if you prepare its own directory under the directory XF86Setup/texts. Please see under the directory XF86Setup/texts/generic.

3.1. Initial questions

If you have an existing XF86Config file, you will be asked if you would like to use it to set the default values of various configuration settings. If you've already got an (at least somewhat) working configuration you will want to do this.

If you are running on an OS which has a mouse driver in the kernel (e.g. SCO or SVR4), you may be asked if you'd like to use it.

Once the questions (if any) are completed, you will see a message indicating that the program is ready to switch into graphics mode. Just press Enter. If you don't get a graphics screen saying *Welcome to XFree86 Setup* within a minute, something has probably hung, you can try pressing Ctrl-Alt-Backspace to switch back to text mode and you'll probably have to use xf86config instead of XF86Setup.

3.2. Configuration areas

Once the VGA16 server is started, and once the program has finished loading, you will see a screen with six buttons along the top and three along the bottom. The buttons along the top correspond to the general categories of configuration settings. They can be done in any order. Each of these areas is explained in detail below. The bottom row consists of the **Abort**, **Done**, and **Help** buttons.

Abort does as it name implies. It exits the program without saving any changes that have been made. The one possible exception is the link to the mouse device. Any change to that is made as soon as **Apply** is selected.

Done should be selected when you've finished configuration in each of the various categories.

The **Help** can be pressed at any time to get on-line help regarding the current configuration screen.

You should start with configuring your mouse as it will make things a lot easier to perform the configuration of other categories.


TOC

3.2.1. Mouse

The mouse configuration screen is used to get the mouse working properly. There are key bindings for everything so that you can easily configure the mouse, if it's not already working.

The screen includes a representation of a white mouse with three buttons. As you move your mouse it should show the pointer coordinates on the mouse and the buttons should turn black as you press the corresponding button on your mouse. If that is not happening, then your mouse is not correctly configured.

Along the top are some rows of buttons corresponding to the various possible protocols. There will also be several buttons and a couple of sliders for other settings, a visual representation of the mouse, and a button to apply any changes. There may also be an entry box in which the device can be set along with a list of possible devices.

First try moving your mouse around and see if the pointer moves correctly. If so, try testing that the buttons are working properly. If those are working as desired, go ahead and go on to another configuration area.

If the mouse pointer doesn't move at all, you need to fix either the mouse device or the protocol (or both). You can press 'n' followed by a Tab, to move to the list of mouse devices and select a different one. Pressing 'p' will pick the next available protocol on the list (protocols that are not available on your OS will be greyed-out). If you have a PnP mouse, it may be easiest to just select "Auto" as the protocol. After changing these, press 'a' to apply the changes and try again. Repeat the process until you are getting some response from your mouse.

If the mouse pointer or button indicators do something when you move the mouse, but the pointer is not moving properly, you probably have the wrong protocol selected. Try with a different one.

Most mice these days use the **Microsoft** protocol, the second most common is **MouseSystems**. Some mice do both. These *dual-protocol* mice have various methods of switching between the two protocols. Some have a switch on the mouse itself. Some are switched by sending a certain signal to the mouse when opening a connection to the mouse. These signals can be controlled by using different combinations of the 'ClearDTR' and 'ClearRTS' settings. Other mice require a button to be depressed when the mouse is opened (when the mouse driver first tries to talk to it). If your mouse uses this method, hold down the appropriate button while selecting apply (pressing 'a').

Once the mouse pointer is moving correctly, test that all three buttons are working properly. If your mouse only has two buttons, select 'Emulate3Buttons' and you should be able to press both buttons simultaneously to emulate the missing middle button. If not all of the buttons are working, try changing the 'ChordMiddle' setting or you may be using a protocol that is similar to that of your mouse, but not quite right.

3.2.2. Keyboard

You need to specify the model and layout of your keyboard (and press apply) if they are not already correct. The graphical representation of the keyboard will be updated when you choose a different model.

For non-U.S. keyboards you may wish to choose a variant from the list (at this time there is only one

available variant: nodeadkeys>).

You can also pick from the options to the right, if you wish.

TOC

3.2.3. Card

Pick your card from the list.

If there are README files that may pertain to your card the 'Read README file' button will then be usable (i.e. not greyed out). Please read them.

If your card is not in the list, or if there are any special settings listed in the README file as required by your card, you can press the 'Detailed Setup' button to make sure that the required settings are selected. Otherwise, you're finished with configuring your card.

To use 'Detailed Setup': First select the appropriate server for your card. Then read the README file corresponding to the selected server by pressing the 'Read README file' button (it won't do anything, if there is no README).

Next, pick the chipset, and Ramdac of your card, if directed by the README file. In most cases, you don't need to select these, as the server will detect (probe) them automatically.

The clockchip should generally be picked, if your card has one, as these are often impossible to probe (the exception is when the clockchip is built into one of the other chips).

Choose whatever options are appropriate (again, according to the README).

You can also set the maximum speed of your Ramdac. Some Ramdacs are available with various speed ratings. The max speed cannot be detected by the server so it will use the speed rating of the slowest version of the specified Ramdac, if you don't specify one.

Additionally, you can also specify the amount of RAM on your card, though the server will usually be able to detect this.

3.2.4. Modeselect

Use this one to pick which depth you prefer to use (this determines how many colors can be displayed at a time) and to select all of the modes you are interested in possibly using.

Your hardware may not be able to support all of depth and mode combinations that can be selected. Any unsupported combinations will automatically be rejected by the server when it tries to startup. Note also that if you select multiple modes, you will get a virtual screen as large as the largest of the usable modes.

3.2.5. Monitor

Enter the horizontal and vertical frequency ranges that your monitor supports in the corresponding entry boxes near the top of the screen. You can enter specific frequencies or ranges of frequencies (separated by hyphens). If the monitor supports several different frequencies or ranges, list them all, separated by commas.

If you can not find this information in you monitor's manual, pick one of the choices from the list of common monitor capabilities. The program will use conservative values for each of these, so you'll get better performance if you type in the correct values from your monitor manual.

3.2.6. Other

You can probably just skip this one.

3.2.7. Completing the configuration

Once you've finished with the above, press the 'Done' button and then the 'Okay' button which will appear. You will then be switched back to text mode.

3.3. Back to text mode

The program will now attempt to start the appropriate server for your card, with all of the configuration settings you selected. If for some reason it is unable to start the server, you have likely selected an improper setting and will be asked if you would like to return to the graphical configuration screen and try again.

3.4. The second server

This is unlikely to happen, but if when the server starts, the display is unreadable, try pressing Ctrl-Alt-+ (using the plus on the numeric keypad) to switch to a different video mode.

The display will show an entry box and three buttons.

The first button allows you to run xvidtune to adjust your video modes. One important point to keep in mind when using xvidtune is that switching video modes with Ctrl-Alt-+ is disabled while xvidtune is running. You must use the 'Next' and 'Prev' buttons to switch modes. Because of this, you should be careful not to move the mouse when pressing either of these. If by some chance the mode you switch to doesn't produce a readable display on your monitor, you can then just press the mouse button again to move to the next (hopefully readable) mode.

The second button causes the settings you've made to be written to the filename given in the entry box. After saving the settings a message will appear indicating that it has finished. Just press the 'Okay' button and you're done.

And the third button causes the program to exit without saving any of the configuration settings.

3.5. Ending text

You are returned to text mode and the program will print a `Configuration complete.' message. You should now have a usable configuration file and can start the X server by whichever method you wish (usually either the 'startx' command or via 'xdm').

Previous: *What to Do - An Overview*
Next: *Running `xf86config`*

TOC

4. Running `xf86config`

From a text screen, run the `xf86config` program. This program should be run as *root* (although not absolutely necessary, it will allow xf86config to do more of the work for you). You can press your interrupt character (usually Control-C or perhaps Delete), at any time to stop the program, if you need to. You can just start it over again.

The `xf86config` program provides instructions on screen as to what you need to do. Following are some notes that document the various stages in the process. They should help you get through the process quickly and provide some documentation for those people who like to know what they're getting themselves into, before running a program.

4.1. The intro screen

First, `xf86config` begins by telling you a few things like the fact that it can help you setup an XF86Config file or that you can do the job yourself with an editor. Just read what it says and press Enter when done.

4.2. Getting your `PATH` right

The program will next check that you have the directory `/usr/X11R6` (the standard installation directory) on your system and tell you that it needs to be in your `PATH` environment variable.

It will also check if you have the `/usr/X386` directory as used by older (pre 3.0) versions of XFree86. If by chance you do, it will warn you that `/usr/X11R6` must be before `/usr/X386` in your `PATH`.

If everything is okay, just press Enter and go on, otherwise press Control-C to exit and make any necessary changes and restart `xf86config`.

4.3. Mouse setup

Pick the mouse type from the menu and enter the name of the device to which mouse is connected, as directed.

If you are using an OS (e.g. SVR4, SCO) that has a built in mouse driver that the Xserver could use, you'll need to edit the XF86Config file to setup your mouse, so just pick any mouse from the list and press enter when asked for the device.

If you don't know which protocol your mouse uses, you'll just have to guess (the xf86config program will give you some hints as to which might be most likely) and then see the troubleshooting section if it doesn't work when you run the server.

The xf86config program has not been updated to allow you to select the latest mouse protocols, so y may have to edit the config file by hand after xf86config has finished.

4.4. Keyboard setup

Simply answer yes to the question regarding keyboard setup.

If there is some reason you need to use the right-alt and control keys for something else, you can enter no.

4.5. Monitor setup

Setting up a monitor consists of entering the specifications of your monitor and a description of the model and manufacturer.

You are first asked for the horizontal sync rate. It is **VERY** important to enter the correct value(s) from the manual. If one of the ranges given matches the rate of your monitor, then pick it, otherwise pick `custom` and enter the values from your manual.

Next is the vertical refresh rate. Again, it is **VERY** important that this parameter be specified correctly. Enter it in a manner similar to the horizontal sync rate.

If either rate is mis-specified, it can result in damage to your monitor.

Finally, you are asked for an "identifier", your monitor manufacturer, and model. You can just press enter to get through these quickly.

4.6. Selecting your card

You are next asked if you would like to view the database of cards. Picking your card from the list will cause the answers to the questions in the next two sections to be filled in for you and so can save a little time.

If your card does not appear in the list, just press `q` and enter to skip on to the next step - where you'll have to answer the questions yourself.

4.7. Server selection

If you selected your card in the previous step, then server selection is easy - just use the recommendation from the database.

If you have a card which uses one of the chipsets for which a specific server exists (Mach8, Mach32, Mach64, AGX/XGA, 8514/A, S3, I128, P9000) you'll want to pick the `accel` option.

Otherwise you'll probably want to use the SVGA server.

Next, answer yes when the program asks if you want it to set the symbolic link for you. If you picked the `accel` option, you'll also need to indicate which particular accelerated server to link to.

4.8. Screen/Video configuration

Pick the appropriate option from the list to indicate the amount of memory on your video card.

Then you are asked to provide and identifier, the manufacturer, and the model of your card. You can just press enter to skip through these, if you wish.

Next, the program will ask for the type of RAMDAC and Clockchip on your card. If your card was in the database, you should just to tell it to use the values from the database.

If you don't have one of the listed RAMDACs or Clockchips on your card, just press enter when asked what type you have. If you do not have a programmable clock chip, the program will next attempt to probe to find out what clock rates are supported by your clock chip.

4.9. Mode Selection

Now you get to tell the program which video modes you would like to be able to run.

The program will show you the common modes that should work with your card (some might not work with your monitor, but if you've correctly specified the monitor's sync rates, the X server will just ignore them when it runs).

You could just accept the settings as they are given, but you'll probably wish to reverse the order. For example, if you have a card with 1 Meg RAM, it will list the modes

```
"640x480" "800x600" "1024x768" for 8bpp
```

Select `1` to change the settings for 8bpp and the type `432` to select the reverse order.

When you've select the modes, in the order you wish, select option `4` to continue.

4.10. Creating the `XF86Config` file

The program will now ask if you would like to write the configuration settings you've selected to the file `XF86Config`. Answer yes.

4.11. Some final notes

Lastly, the program tells you that it's finished its part of this process and counsels you to check the file before using it. The next section covers the changes that are most likely to be needed.

Quick-Start Guide to XFree86 Setup : *Running `xf86config`*
Previous: *Using `XF86Setup`*
Next: *Fixing the XF86Config file*

5. Fixing the XF86Config file

Use an editor to look at the `XF86Config` file. Here are some things that may need to be changed:

- If you are running an operating system which has built-in mouse support, you'll want to change the `Pointer` section. Specifically, you should set the `Protocol` to `OSMouse` (SCO) or `Xqueue` (SVR4, some SVR3) and you should remove the `Device` line.
- If you are running a system with the Xqueue event driver and would like to use it, change the `Protocol` setting in the `Keyboard` section to `Xqueue`.

Once you are satisfied that the configuration is correct, copy the XF86Config file to `/usr/X11R6/lib/X11` and run the 'startx' command.

You should now have a running X server. If it's running but the display doesn't look as good as you think it should (i.e. it doesn't fill the whole screen, it's off-center, it's wrapping around on one side, etc.) see the section on `xvidtune`. If there is some other problem, see the troubleshooting section.

6. Running `xvidtune`

If you need to make adjustments to the video display, `xvidtune` is the tool to use.

Simply enter the command `xvidtune` from a shell prompt within an xterm. Read the warning and click on `OK`. Next click on the `Auto` button.

Now click on whatever combination of `Up/Down/Left/Right` `Shorter/Taller/Wider/Narrower` is need to adjust the display to your liking.

If you are using a recent S3-based card there will be some extra buttons and entries at the bottom (InvertVCLK, EarlySC, and Blank Delays). These can help solve problems of the display wrapping around a few pixels.

Once the display has been adjusted properly, press the `show` button to printout the correct `ModeLine` to put in the `XF86Config` to make the server always use the current display settings. To aid in copying this information to your XF86Config file, the modeline is also made the current selection allowing you to just paste it into your editor.

If you would like to adjust your other modes, you can click on the `Next` and `Prev` buttons to switch modes.

When you are through using `xvidtune` simply press on the `Quit` button.

7. Troubleshooting

Since you're reading this, something must not have gone the way you had hoped (or else you just really enjoy reading).

Below are listed some common problems that may occur during configuration and some hints for solving them. However, there are just too many different combinations of hardware and software configurations, and, well, just too many things that can go wrong, for this document and the tools it documents, to cover every case.

If after trying the steps in the previous sections and checking the hints in this section, you still are unable to get your system working, you'll have to read the full documentation. Read the README file for your card and OS (if they exist), the XFree86 Configuration Guide (README.Config), and the XF86Config man page.

You should also look at the XFree86 FAQ for more up-to-date information, especially if you are trying to configure a fairly new card.

If all else fails, you can try posting a message to comp.windows.x.i386unix or comp.os.linux.x or send email to XFree86@XFree86.org.

7.1. The mouse doesn't move correctly, it stays in one area of the screen

You've selected the wrong protocol for your mouse. Try a different one.

7.2. The server doesn't start, it says the mouse is busy.

Well, it's probably right. This most often happens on Linux systems that have `gpm` running. Kill the `gpm` process and try `startx` again.

7.3. The middle button doesn't work.

There's no easy answer to this one. It's a lot of trial and error. You need to make sure you're running the right protocol for your mouse.

Many three button mice are "dual protocol" which means that they have both a 2-button and 3-button mode. The way to get the mouse to switch into 3-button mode (which usually then uses MouseSystems protocol) varies between different models.

You may need to slide a switch on the mouse or hold down the middle button when starting the server. Other methods of switching modes can be done by the server, you just have to find the right combination of settings for your mouse. See the Pointer section of the XF86Config man page for a complete list of settings.

7.4. The display is shifted to the left/right/top/bottom

See the section on xvidtune.

7.5. I don't appear to have xf86config or xvidtune on my system

Hmmm. A couple of possibilities:

1. Your `PATH` is not set correctly. Make sure it includes the bin directory for the XFree86 binaries (usually, `/usr/X11R6/bin`

2. You don't have a complete installation of XFree86. Go back to wherever you got XFree86 and get the missing pieces.

TOC

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/QStart.sgml,v 3.4.2.3 1998/04/29
04:18:31 dawes Exp $
```

Quick-Start Guide to XFree86 Setup : *Troubleshooting*
Previous: *Running `xvidtune`*
Next: *Quick-Start Guide to XFree86 Setup*

README for XFree86[tm] 3.3.5

TOC

The XFree86 Project, Inc

17 August 1999

XFree86 is a port of X11R6.3 that supports several Unix and Unix-like operating systems on Intel and other platforms. This release is a quick update to XFree86 3.3.4, fixing a few bugs that were found after releasing it and adding support to two more operating systems. The release is available as source patches against the X Consortium X11R6.3 code and the XFree86 3.3.4 release. Binary distributions for many architectures are also available.

---

1. What's new in XFree86 3.3.5

2. Systems XFree86 has been tested on

3. Supported video-card chip-sets

4. Where to get more information

5. Credits

6. Contact information

7. The XFree86 Project, Inc.

8. Source and binary archive sites

TOC

1. What's new in XFree86 3.3.5

For a summary of new features in this release, please refer to the RELNOTES file. For a detailed list of changes, refer to the CHANGELOG file in the source distribution.

2. Systems XFree86 has been tested on

Note: Not all systems listed here have been tested with the current release.

SVR4.0:

- Esix: 4.0.3A, 4.0.4, 4.0.4.1
- Microport: 2.2, 3.1, 4.1, 4.2
- Dell: 2.1, 2.2, 2.2.1
- UHC: 2.0, 3.6
- Consensys: 1.2
- MST: 4.0.3 (Load 2.07 and Load 3.02)
- ISC: 4.0.3
- AT&T: 2.1, 4.0
- NCR: MP-RAS
- SunSoft: Solaris x86 2.1, 2.4, 2.5, 2.5.1, 2.6
- PANIX 5.0 for AT

SVR4.2:

- Consensys
- Novell/SCO UnixWare

SVR3:

- ISC: 3.0, 4.0, 4.1

Others:

- NetBSD 1.0, 1.1, 1.2, 1.2.1, 1.3, 1.3.1, 1.3.2, 1.3.3, 1.4
- OpenBSD 2.0, 2.1, 2.2, 2.3, 2.4
- FreeBSD 2.0.5, 2.1, 2.1.5, 2.1.6, 2.1.7, 2.1.7.1, 2.2, 2.2.1, 2.2.2, 2.2.5, 2.2.6, 2,2,7, 2.2.8, 3.0, 3.1, 3.2
- Linux (Intel x86, DEC Alpha/AXP and m68k)
- LynxOS x86 2.3.0, 2.4.0, 2.5.x, 3.0.x
- LynxOS microSPARC 2.4.0, 2.5.x, 3.0.x
- LynxOS PowerPC 2.4.0, 2.5.x, 3.0.x
- OS/2 Warp 3 FP5/17/22, Warp 4 -/FP1

PC98:

- FreeBSD(98) 2.0.5, 2.1, 2.1.5, 2.1.7.1, 2.2, 2.2.1, 2.2.2, 2.2.5, 2.2.6, 2,2,7, 2.2.8, 3.0, 3.1, 3.2
- NetBSD/pc98 (based on NetBSD 1.2, 1.2.1, 1.3, 1.3.1, 1.3.2, 1.3.3)

- PANIX 5.0 for 98
- Linux/98

TOC

README for XFree86[tm] 3.3.5 **:** *Systems XFree86 has been tested on*

Previous: *What's new in XFree86 3.3.5*

Next: *Supported video-card chip-sets*

3. Supported video-card chip-sets

At this time, XFree86 3.3.5 supports the following chipsets:

Ark Logic

ARK1000PV, ARK1000VL, ARK2000PV, ARK2000MT

Alliance

AP6422, AT24

ATI

18800, 18800-1, 28800-2, 28800-4, 28800-5, 28800-6, 68800-3, 68800-6, 68800AX, 68800LX, 88800GX-C, 88800GX-D, 88800GX-E, 88800GX-F, 88800CX, 264CT, 264ET, 264VT, 264GT, 264VT-B, 264VT3, 264GT-B, 264GT3 (this list includes the Mach8, Mach32, Mach64, 3D Rage, 3D Rage II and 3D Rage Pro)

Avance Logic

ALG2101, ALG2228, ALG2301, ALG2302, ALG2308, ALG2401

Chips & Technologies

65520, 65525, 65530, 65535, 65540, 65545, 65546, 65548, 65550, 65554, 65555, 68554, 69000, 64200, 64300

Cirrus Logic

CLGD5420, CLGD5422, CLGD5424, CLGD5426, CLGD5428, CLGD5429, CLGD5430, CLGD5434, CLGD5436, CLGD5440, CLGD5446, CLGD5462, CLGD5464, CLGD5465, CLGD5480, CLGD6205, CLGD6215, CLGD6225, CLGD6235, CLGD6410, CLGD6412, CLGD6420, CLGD6440, CLGD7541(*), CLGD7543(*), CLGD7548(*), CLGD7555(*)

Cyrix

MediaGX, MediaGXm

Compaq

AVGA

Digital Equipment Corporation

TGA

Epson

SPC8110

Genoa


TOC

GVGA

IBM

8514/A (and true clones), XGA-2

Intel

i740

IIT

AGX-014, AGX-015, AGX-016

Matrox

MGA2064W (Millennium), MGA1064SG (Mystique and Mystique 220), MGA2164W (Millennium II PCI and AGP), G100, G200, G400

MX

MX68000(*), MX680010(*)

NCR

77C22(*), 77C22E(*), 77C22E+(*)

NeoMagic

2200, 2160, 2097, 2093, 2090, 2070

Number Nine

I128 (series I, II and IV), Revolution 3D (T2R)

NVidia/SGS Thomson

NV1, STG2000, RIVA128, Riva TNT, Riva TNT2

OAK

OTI067, OTI077, OTI087

RealTek

RTG3106(*)

Rendition

V1000, V2x00

S3

86C911, 86C924, 86C801, 86C805, 86C805i, 86C928, 86C864, 86C964, 86C732, 86C764, 86C765, 86C767, 86C775, 86C785, 86C868, 86C968, 86C325, 86C357, 86C362, 86C375, 86C375, 86C385, 86C988, 86CM65, 86C260

SiS

86C201, 86C202, 86C205, 86C215, 86C225, 5597, 5598, 6326, 530, 620

3dfx


TOC

Voodoo Banshee, Voodoo3

3DLabs

GLINT 500TX, GLINT MX, Permedia, Permedia 2, Permedia 2v

Tseng

ET3000, ET4000AX, ET4000/W32, ET4000/W32i, ET4000/W32p, ET6000, ET6100

Trident

TVGA8800CS, TVGA8900B, TVGA8900C, TVGA8900CL, TVGA9000, TVGA9000i, TVGA9100B, TVGA9200CXR, Cyber9320(*), TVGA9400CXi, TVGA9420, TGUI9420DGi, TGUI9430DGi, TGUI9440AGi, TGUI9660XGi, TGUI9680, ProVidia 9682, ProVidia 9685(*), Cyber 9382, Cyber 9385, Cyber 9388, 3DImage975, 3DImage985, Cyber 9397, Cyber 9520, Cyber 9525, Blade3D, CyberBlade

Video 7/Headland Technologies

HT216-32(*)

Weitek

P9000, P9100

Western Digital/Paradise

PVGA1

Western Digital

WD90C00, WD90C10, WD90C11, WD90C24, WD90C24A, WD90C30, WD90C31, WD90C33

(*) Note, chips marked in this way have either limited support or the drivers for them are not actively maintained.

All of the above are supported in both 256 color, and some are supported in mono and 16 color modes, and some are supported an higher color depths.

Refer to the chipset-specific `README` files (currently for TGA, Matrox, Mach32, Mach64, NVidia, Oak, P9000, S3 (except ViRGE), S3 ViRGE, SiS, Video7, Western Digital, Tseng (W32), Tseng (all), AGX/XGA, ARK, ATI (SVGA server), Chips and Technologies, Cirrus, Trident, NeoMagic, Rendition, Epson, 3DLabs) i740) for more information about using those chipsets.

The monochrome server also supports generic VGA cards, using 64k of video memory in a single bank, the Hercules monochrome card, the Hyundai HGC1280, Sigma LaserView, Visa and Apollo monochrome cards.

The VGA16 server supports memory banking with the ET4000, Trident, ATI, NCR, OAK and Cirrus 6420 chipsets allowing virtual display sizes up to about 1600x1200 (with 1MB of video memory). For other chipsets the display size is limited to approximately 800x600.

Next: *Where to get more information*

TOC

4. Where to get more information

Additional documentation is available in the *XFree86(1)*, *XF86Config(4/5)*, *XF86_SVGA(1)*, *XF86_Mono(1)*, *XF86_VGA16(1)*, *XF86_Accel(1)*, *XF86Setup(1)* and *xvidtune(1)* manual pages. In addition, several `README` files and tutorial documents are provided. These are available in `/usr/X11R6/lib/X11/doc` in the binary distributions, and in `xc/programs/Xserver/hw/xfree86/doc` in the source distribution.

The files QuickStart.doc and README.Config should be consulted for information on how to set up the XFree86 servers. All supplied documents, manual pages, and the XFree86 FAQ should be read before contacting the XFree86 team for assistance.

Documentation on SVGA driver development can be found in the directory `/usr/X11R6/lib/Server/VGADriverDoc` in the binary distribution, and in the directory `xc/programs/Xserver/hw/xfree86/VGADriverDoc` in the source distribution.

If you are totally at a loss, you can contact the XFree86 Support Team at *<XFree86@XFree86.Org>*. Before doing so, please make sure that you are using the latest release of XFree86. Check the versions listed on ftp://ftp.xfree86.org/pub/XFree86.

There is a Usenet news group comp.windows.x.i386unix that contains mostly discussions about XFree86 and related topics. Many questions can be answered there.

5. Credits

XFree86 was originally put together by:

- David Dawes <*dawes@XFree86.org*>
- Glenn Lai <*glenn@cs.utexas.edu*>
- Jim Tsillas <*jtsilla@ccs.neu.edu*>
- David Wexelblat <*dwex@XFree86.org*>

XFree86 support was integrated into the base X11R6 distribution by:

- Stuart Anderson <*anderson@metrolink.com*>
- Doug Anson <*danson@lgc.com*>
- Gertjan Akkerman <*akkerman@dutiba.twi.tudelft.nl*>
- Mike Bernson <*mike@mbsun.mlb.org*>
- Robin Cutshaw <*robin@XFree86.org*>
- David Dawes <*dawes@XFree86.org*>
- Marc Evans <*marc@XFree86.org*>
- Pascal Haible <*haible@izfm.uni-stuttgart.de*>
- Matthieu Herrb <*Matthieu.Herrb@laas.fr*>
- Dirk Hohndel <*hohndel@XFree86.org*>
- David Holland <*davidh@use.com*>
- Alan Hourihane <*alanh@fairlite.demon.co.uk*>
- Jeffrey Hsu <*hsu@soda.berkeley.edu*>
- Glenn Lai <*glenn@cs.utexas.edu*>
- Ted Lemon <*mellon@ncd.com*>
- Rich Murphey <*rich@XFree86.org*>
- Hans Nasten <*nasten@everyware.se*>
- Mark Snitily <*mark@sgcs.com*>
- Randy Terbush <*randyt@cse.unl.edu*>
- Jon Tombs <*tombs@XFree86.org*>
- Kees Verstoep <*versto@cs.vu.nl*>
- Paul Vixie <*paul@vix.com*>
- Mark Weaver <*Mark_Weaver@brown.edu*>
- David Wexelblat <*dwex@XFree86.org*>


TOC

- Philip Wheatley <*Philip.Wheatley@ColumbiaSC.NCR.COM*>
- Thomas Wolfram <*wolf@prz.tu-berlin.de*>
- Orest Zborowski <*orestz@eskimo.com*>

386BSD, FreeBSD, NetBSD support by:

- Rich Murphey <*Rich@XFree86.org*>

NetBSD, OpenBSD support by:

- Matthieu Herrb <*Matthieu.Herrb.@laas.fr*>

Original 386BSD port by:

- Pace Willison,
- Amancio Hasty Jr <*hasty@netcom.com*>

Mach 386 support by:

- Robert Baron <*Robert.Baron@ernst.mach.cs.cmu.edu*>

Linux support by:

- Orest Zborowski <*orestz@eskimo.com*>

DG/ux support by:

- Takis Psarogiannakopoulos <*takis@dpmms.cam.ac.uk*>

SCO Unix support by:

- David McCullough <*davidm@stallion.oz.au*>

Amoeba support by:

- Kees Verstoep <*versto@cs.vu.nl*>

Minix-386 support by:

- Philip Homburg <*philip@cs.vu.nl*>

OSF/1 support by:

- Marc Evans <*Marc@XFree86.org*>

BSD/OS support by:

- Hans Nasten <*nasten@everyware.se*>,
- Paul Vixie <*paul@vix.com*>

Solaris support by:

- Doug Anson <*danson@lgc.com*>,
- David Holland <*davidh@use.com*>

ISC SVR3 support by:

- Michael Rohleder <*michael.rohleder@stadt-frankfurt.de*>

LynxOS support by:

- Thomas Mueller <*tmueller@sysgo.de*>

OS/2 support by:

- Holger Veit <*Holger.Veit@gmd.de*>
- Sebastien Marineau <*s521936@aix1.uottawa.ca*>

TOC

Linux shared libraries by:

- Orest Zborowski <*orestz@eskimo.com*>,
- Dirk Hohndel <*hohndel@XFree86.org*>

PC98 support by:

- Toyonori Fujiura <*toyo@ibbsal.or.jp*>,
- Hiroyuki Aizu <*aizu@jaist.ac.jp*>,
- Tetsuya Kakefuda <*kakefuda@tag.iijnet.or.jp*>,
- Takefumi Tsukada <*tsuka@linkt.imasy.or.jp*>,
- H.Komatsuzaki,
- Naoki Katsurakawa <*katsura@prc.tsukuba.ac.jp*>,
- Shuichiro Urata <*s-urata@nmit.tmg.nec.co.jp*>,
- Yasuyuki Kato <*yasuyuki@acaets0.anritsu.co.jp*>,
- Michio Jinbo <*karl@spnet.ne.jp*>,
- Tatsuya Koike <*koiket@focus.rim.or.jp*>,
- Koichiro Suzuki <*s-koichi@nims.nec.co.jp*>,
- Tsuyoshi Tamaki <*tamaki@sail.t.u-tokyo.ac.jp*>,
- Isao Ohishi <*ohishi@hf.rim.or.jp*>,
- Kohji Ohishi <*atena@njk.co.jp*>,
- Shin'ichi Yairo <*QZR00522@nifty.ne.jp*>,
- Kazuo Ito <*ft4k-itu@asahi-net.or.jp*>,
- Jun Sakuma <*i931361@jks.is.tsukuba.ac.jp*>,
- Shuichi Ueno <*uenos@ppp.bekkoame.or.jp*>,
- Ishida Kazuo <*ishidakz@obp.cl.nec.co.jp*>,
- Takaaki Nomura <*amadeus@yk.rim.or.jp*>,
- Tadaaki Nagao <*nagao@cs.titech.ac.jp*>,
- Minoru Noda <*mnoda@cv.tottori-u.ac.jp*>,
- Naofumi Honda <*honda@Kururu.math.hokudai.ac.jp*>,
- Akio Morita <*amorita@bird.scphys.kyoto-u.ac.jp*>,
- Takashi Sakamoto <*sakamoto@yajima.kuis.kyoto-u.ac.jp*>,
- Yasuhiro Ichikawa <*cs94006@mbox.sist.ac.jp*>,
- Kazunori Ueno <*jagarl@creator.club.or.jp*>,
- Yasushi Suzuki <*suz@d2.bs1.fc.nec.co.jp*>,
- Satoshi Kimura <*KFB03633@nifty.ne.jp*>,
- Kazuhiko Uno <*Kazuhiko.Uno@softvision.co.jp*>,
- Tomiharu Takigami <*takigami@elsd.mt.nec.co.jp*>,
- Tomomi Suzuki <*suzuki@grelot.elec.ryukoku.ac.jp*>,
- Toshihiko Yagi <*j2297222@ed.kagu.sut.ac.jp*>,

TOC

- Masato Yoshida (Contributor of PW805i support)

Original accelerated code by:

- Kevin E. Martin <*martin@cs.unc.edu*>,
- Rik Faith <*faith@cs.unc.edu*>,
- Jon Tombs <*tombs@XFree86.org*>

XFree86 Acceleration Architecture (XAA) by:

- Harm Hanemaayer <*H.Hanemaayer@inter.nl.net*>,

S3 accelerated code by:

- Jon Tombs <*tombs@XFree86.org*>,
- Harald Koenig <*koenig@tat.physik.uni-tuebingen.de*>,
- David Wexelblat <*dwex@XFree86.org*>,
- David Dawes <*dawes@XFree86.org*>,
- Robin Cutshaw <*robin@XFree86.org*>,
- Amancio Hasty <*hasty@netcom.com*>,
- Norbert Distler <*Norbert.Distler@physik.tu-muenchen.de*>,
- Leonard N. Zubkoff <*lnz@dandelion.com*>,
- Bernhard Bender <*br@elsa.mhs.compuserve.com*>,
- Dirk Hohndel <*hohndel@XFree86.org*>,
- Joe Moss <*joe@XFree86.org*>

S3V accelerated code by:

- Harald Koenig <*koenig@tat.physik.uni-tuebingen.de*>,
- Kevin Brosius <*Cobra@compuserve.com*>
- Berry Dijk <*berry_dijk@tasking.nl*>
- Dirk Hohndel <*hohndel@XFree86.org*>
- Huver Hu <*huver@amgraf.com*>
- Dirk Vangestel <*gesteld@sh.bel.alcatel.be*>

Mach32 accelerated code by:

- Kevin E. Martin <*martin@cs.unc.edu*>,
- Rik Faith <*faith@cs.unc.edu*>,
- Mike Bernson <*mike@mbsun.mlb.org*>,
- Mark Weaver <*Mark_Weaver@brown.edu*>,
- Craig Groeschel <*craig@metrolink.com*>
- Bryan Feir <*jenora@istar.ca*>

Mach64 accelerated code by:

- Kevin E. Martin <*martin@cs.unc.edu*>,

Mach8, 8514 accelerated code by:

- Kevin E. Martin <*martin@cs.unc.edu*>,


TOC

- Rik Faith <*faith@cs.unc.edu*>,
- Tiago Gons <*tiago@comosjn.hobby.nl*>,
- Hans Nasten <*nasten@everyware.se*>,
- Scott Laird <*scott@laird.com*>

Cirrus accelerated code by:

- Simon Cooper <*scooper@vizlab.rutgers.edu*>,
- Harm Hanemaayer <*H.Hanemaayer@inter.nl.net*>,
- Bill Reynolds <*bill@goshawk.lanl.gov*>,
- Corin Anderson <*corina@the4cs.com*>

Western Digital accelerated code by:

- Mike Tierney <*floyd@pepsi.eng.umd.edu*>,
- Bill Conn <*conn@bnr.ca*>

P9000 accelerated code by:

- Erik Nygren <*nygren@mit.edu*>,
- Harry Langenbacher <*harry@brain.jpl.nasa.gov*>
- Chris Mason <*mason@mail.csh.rit.edu*>
- Henrik Harmsen <*harmsen@eritel.se*>

AGX accelerated code by:

- Henry Worth <*haworth@wco.com*>,

Number Nine I128 driver by:

- Robin Cutshaw <*robin@XFree86.org*>,

ET4000/W32 accelerated code by:

- Glenn Lai <*glenn@cs.utexas.edu*>,

ET6000 SVGA and accelerated support (both based on the existing W32 code) by:

- Koen Gadeyne <*koen.gadeyne@barco.com*>,

Oak Technologies Inc. accelerated code by:

- Jorge Delgado <*ernar@dit.upm.es*>,

16 color VGA server by:

- Gertjan Akkerman <*akkerman@dutiba.twi.tudelft.nl*>

2 color VGA and non-VGA mono servers by:

- Pascal Haible <*haible@izfm.uni-stuttgart.de*>

ATI SVGA driver by:

- Per Lindqvist <*pgd@compuram.bbt.se*> and Doug Evans <*dje@cygnus.com*>.
- Ported to X11R5 by Rik Faith <*faith@cs.unc.edu*>.
- Rewritten by Marc Aurele La France <*tsi@ualberta.ca*>

WD90C24 support by:

- Brad Bosch <*brad@lachman.com*>

TOC

Trident SVGA driver by:

- Alan Hourihane <*alanh@fairlite.demon.co.uk*>

SiS SVGA driver by:

- Alan Hourihane <*alanh@fairlite.demon.co.uk*>
- Xavier Ducoin <*xavier@rd.lectra.fr*>
- Dirk Hohndel <*hohndel@XFree86.Org*>

DEC 21030 (TGA) server by:

- Alan Hourihane <*alanh@fairlite.demon.co.uk*>
- Harald Koenig <*koenig@tat.physik.uni-tuebingen.de*>

NCR SVGA driver by:

- Stuart Anderson <*anderson@metrolink.com*> with the permission of NCR Corporation

Cirrus SVGA driver by:

- Bill Reynolds <*bill@goshawk.lanl.gov*>,
- Hank Dietz <*hankd@ecn.purdue.edu*>,
- Simon Cooper <*scooper@vizlab.rutgers.edu*>,
- Harm Hanemaayer <*H.Hanemaayer@inter.nl.net*>,
- Corin Anderson <*corina@the4cs.com*>

Cirrus CL64xx driver by:

- Manfred Brands <*mb@oceonics.nl*>
- Randy Hendry <*randy@sgi.com*>
- Jeff Kirk <*jeff@bambam.dsd.ES.COM*>

Compaq SVGA driver by:

- Hans Oey <*hans@mo.hobby.nl*>
- Ming Yu <*yum@itp.ac.cn*>
- Gerry Toll <*gtoll@tc.cornell.edu*>

Oak SVGA driver by:

- Steve Goldman <*sgoldman@encore.com*>
- Jorge Delgado <*ernar@dit.upm.es*>

ARK Logic SVGA driver by:

- Harm Hanemaayer <*H.Hanemaayer@inter.nl.net*>
- Leon Bottou <*bottou@laforia.ibp.fr*>

AL2101 SVGA driver by:

- Paolo Severini <*lendl@dist.dist.unige.it*>

Avance Logic ``ali'' SVGA driver by:

- Ching-Tai Chiu <*cchiu@netcom.com*>

Chips & Technologies SVGA driver by:

- Regis Cridlig <*cridlig@dmi.ens.fr*>


TOC

- Jon Block <*block@frc.com*>
- Mike Hollick <*hollick@graphics.cis.upenn.edu*>
- Nozomi Ytow
- Egbert Eich <*Egbert.Eich@Physik.TH-Darmstadt.DE*>
- David Bateman <*dbateman@ee.uts.edu.au*>
- Xavier Ducoin <*xavier@rd.lectra.fr*>

MX SVGA driver by:

- Frank Dikker <*dikker@cs.utwente.nl*>

Video7 SVGA driver by:

- Craig Struble <*cstruble@acm.vt.edu*>

RealTek SVGA driver by:

- Peter Trattler <*peter@sbox.tu-graz.ac.at*>

Apollo Mono driver by:

- Hamish Coleman <*hamish@zot.apana.org.au*>

Matrox SVGA driver by:

- Guy Desbief <*g.desbief@aix.pacwan.net*>
- Radoslaw Kapitan <*kapitan@student.uci.agh.edu.pl*>
- Andrew Vanderstock <*vanderaj@mail2.svhm.org.au*>
- Angsar Hockmann <*Ansgar.Hockmann@hrz.uni-dortmund.de*>
- Michael Will <*Michael.Will@student.uni-tuebingen.de*>
- Andrew Mileski <*aem@ott.hookup.net*>
- Stephen Pitts <*pitts2@memphisonline.com*>
- Dirk Hohndel <*hohndel@XFree86.Org*>
- Leonard N. Zubkoff <*lnz@dandelion.com*>

ViRGE SVGA driver by:

- Sebastien Marineau <*marineau@genie.uottawa.ca*>,
- Harald Koenig <*koenig@tat.physik.uni-tuebingen.de*>

Linux/m68k Frame Buffer Device driver by:

- Martin Schaller
- Geert Uytterhoeven <*Geert.Uytterhoeven@cs.kuleuven.ac.be*>
- Andreas Schwab <*schwab@issan.informatik.uni-dortmund.de*>
- Guenther Kelleter <*guenther@Pool.Informatik.RWTH-Aachen.de*>

Tseng ET4000 and ET6000 SVGA driver by:

- [Unknown authors]
- Dirk Hohndel <*hohndel@XFree86.Org*>
- Koen Gadeyne <*koen.gadeyne@barco.com*>
- ... and others

TOC

P9100 accelerated code by:

- Joerg Knura <*knura@imst.de*>

Rendition code by:

- Tim Rowley <*tor@cs.brown.edu*>
- Marc Langenbach <*mlangen@studcs.uni-sb.de*>

Cyrix accelerated code by:

- Annius Groenink <*Annius.Groenink@cwi.nl*>
- Dirk Hohndel <*hohndel@XFree86.Org*>

Epson code by:

- Thomas Mueller <*tmueller@sysgo.de*>

3DLabs accelerated code by:

- Alan Hourihane <*alanh@fairlite.demon.co.uk*>
- Dirk Hohndel <*hohndel@XFree86.Org*>
- Stefan Dirsch <*sndirsch@suse.de*>
- Helmut Fahrion <*hf@suse.de*>

3dfx accelerated code by:

- Daryll Strauss <*daryll@harlot.rb.ca.us*>
- Scott Bertin

Intel i740 accelerated code by:

- Kevin E. Martin <*martin@cs.unc.edu*>,

XFree86-VidModeExtension and xvidtune client by:

- Kaleb S. Keithley <*kaleb@x.org*>
- David Dawes <*dawes@XFree86.org*>
- Jon Tombs <*tombs@XFree86.org*>
- Joe Moss <*joe@XFree86.org*>

XFree86-Misc extension by:

- Joe Moss <*joe@XFree86.org*>
- David Dawes <*dawes@XFree86.org*>

XFree86-DGA extension by:

- Jon Tombs <*tombs@XFree86.org*>
- Mark Vojkovich <*mvojkovi@ucsd.edu*>
- Harm Hanemaayer <*H.Hanemaayer@inter.nl.net*>,
- David Dawes <*dawes@XFree86.org*>

XInput integration, devices and clients by:

- Frederic Lepied <*lepied@XFree86.Org*> (XInput integration, Wacom tablet, Joystick and extended mouse devices, xsetpointer and xsetmode clients)
- Patrick Lecoanet <*lecoanet@cena.dgac.fr*> (Elographics touchscreen device)


TOC

- Steven Lang <*tiger@tyger.org*> (Summagraphics tablet device)

Other contributors:

- Joerg Wunsch <*joerg_wunsch@uriah.sax.de*> (ET3000 banked mono),
- Thomas Dickey <*dickey@clark.net*> (xterm "new" model ANSI colors and VT220, VT52 emulation).
- Eric Raymond <*esr@snark.thyrsus.com*> (new video mode documentation),
- and an entire horde of beta-testers around the world!

README for XFree86[tm] 3.3.5 **:** *Credits*
Previous: *Where to get more information*
Next: *Contact information*

6. Contact information

Ongoing development planning and support is coordinated by the XFree86 Core Team. At this time the Core Team consists of (in alphabetical order):

- Robin Cutshaw <*robin@XFree86.org*>
- David Dawes <*dawes@XFree86.org*>
- Marc Evans <*marc@XFree86.org*>
- Harm Hanemaayer <*H.Hanemaayer@inter.nl.net*>
- Dirk Hohndel <*hohndel@XFree86.org*>
- Harald Koenig <*koenig@XFree86.org*>
- Rich Murphey <*rich@XFree86.org*>
- Takaaki Nomura <*nomura@XFree86.org*>
- Jon Tombs <*tombs@XFree86.org*>
- David Wexelblat <*dwex@XFree86.org*>

Mail sent to <*Core@XFree86.org*> will reach the core team. Please note that support questions should be sent to <*XFree86@XFree86.org*>.

7. The XFree86 Project, Inc.

The XFree86 Project, Inc, was founded to accomplish two major goals:

1. To provide a vehicle by which XFree86 can be represented in X Consortium, Inc, the organization responsible for the design, development, and release of The X Window System.
2. To provide some basic funding for acquisition of facilities for ongoing XFree86 development, largely to consist of new video hardware and basic computing facilities.

The first of these was the primary motivation. We have held discussions with the X Consortium on and off for many months, attempting to find an avenue by which our loosely-organized free software project could be given a voice within the X Consortium. The bylaws of the Consortium would not recognize such an organization. After an initial investigation about funding, we decided to form our own corporation to provide the avenue we needed to meet the requirements of the X Consortium bylaws.

By doing this, we were able to be involved in the beta-test interval for X11R6, and have contributed the majority of XFree86 to the X11R6 and X11R6.1 core release. The version of XFree86 in the initial X11R6 core is 3.0. The version of XFree86 in the current X11R6.3 release is 3.2.

An additional benefit of this incorporation is that The XFree86 Project, Inc has obtained outside financial support for our work. This will hopefully give us the freedom to be more pro-active in obtaining new video hardware, and enable us to release better products more quickly, as we will be able to go and get what we need, and get it into the hands of the people who can do the work.

The current Board of Directors and Officers of the The XFree86 Project, Inc, are:

- David Dawes, President and Secretary
- Dirk Hohndel, Vice-President
- Glenn Lai, Director
- Rich Murphey, Treasurer
- Jim Tsillas, Director
- Jon Tombs, Director
- David Wexelblat, Director

Email to <*BOD@XFree86.org*> reaches the board of directors.

Our bylaws have been crafted in such a way to ensure that XFree86 is and always will be a free software project. There is no personal financial benefit to any member of the Core Team or any other XFree86 participant. All assets of the corporation remain with the corporation, and, in the event of the dissolution of the corporation, all assets will be turned over to the X Consortium, Inc. It is hoped that by doing this, our corporation will be merely a formalization of what we have been doing in the past, rather than something entirely new.

As of March 1997, The XFree86 Project has revised its source/binary access and release policy. The main points of the new policy are:

- There will be no more time-limited public binary-only beta releases. Instead we plan to increase the frequency of full public releases to about four releases per year.
- The source access/use is divided into three categories:
  - End users. End users have access to only the source of full public releases. The main reason for this restriction is that our development code often contains code from other sources which cannot be released to the public immediately.
  - Active developers (members of the XFree86 ``developer team''). Active developers must formally become non-voting members of the XFree86 Project, and have full access to our internal development source. They are permitted to make time-limited binaries (in coordination with the Core Team) of the servers they are actively working on available to external testers for specific testing.
  - Commercial members. Commercial members are non-voting members of The XFree86 Project who donate US$5000/year to the Project. Additionally, companies who contribute significantly to the development effort of XFree86 can be awarded commercial membership by the Core Team on a yearly bases. Commercial members can use the internal XFree86 development source for derived binary-only products providing that they take full responsibility for supporting the product, and don't call it ``XFree86'' (although the derivation of the product must be acknowledged in any accompanying documentation). Binary packages for the OSs we support which are simply compiled from our internal source without significant added value are explicitly NOT allowed.

Here is a list of the organizations and individuals who have provided sponsorship to The XFree86 Project, Inc, either by financial contribution or by the donation of equipment and resources. The XFree86 Project, Inc gratefully acknowledges these contributions, and hopes that we can do justice to them by continuing to release high-quality free software for the betterment of the Internet community as a whole.

- UUNET Communications Services, Inc.

UUNET Communications Services, Inc, deserves special mention. This organization stepped forward and contributed the entire 1994 X Consortium membership fee on a moment's notice. This single act ensured XFree86's involvement in X11R6.

- GUUG -- 1st German Linux Congress

Also deserving of special mention are the organizers and attendees of the 1st German Linux Congress in Heidelberg. Significant funding to The XFree86 Project has been provided from its proceeds.

- AIB Software Corporation, Herndon, VA
- Roland Alder, Armin Fessler, Patrick Seemann, Martin Wunderli
- American Micro Group
- ATI Technologies Inc
- Andrew Burgess
- Berkeley Software Design, Inc, Colorado Springs, CO
- Caldera, Inc.

TOC

- Delix Computer GmbH, Stuttgart, Germany
- The Destek Group, Inc., Nashua, NH (formerly Synergytics)
- Diamond Multimedia Systems, Inc.
- Digital Equipment Corporation
- Elsa GmbH, Aachen, Germany
- Genoa Systems Corporation
- Helius, Inc.
- Hercules Computer Technology, Inc.
- Ralf Hockens
- Dirk Hohndel
- InfoMagic, Flagstaff, AZ
- Daniel Kraemer
- Epoch Networks, Inc., Irvine, CA
- Frank & Paige McCormick
- Internet Labs, Inc.
- Linux International
- Linux Support Team, Erlangen, Germany
- LunetIX Softfair, Berlin, Germany
- Morse Telecommunications, Long Beach, NY
- MELCO, Inc
- MIRO Computer Products AG, Braunschweig, Germany
- Rich & Amy Murphey
- NCR Corp
- Brett Neumeier
- Number Nine, Lexington, MA
- Kazuyuki Okamoto, Japan
- Prime Time Freeware, San Bruno, CA
- Red Hat Software, Chapel Hill, NC
- Norbert Reithinger
- SPEA Software AG, Starnberg, Germany
- STB Systems
- Clifford M Stein
- Joel Storm
- S.u.S.E. GmbH, Fuerth, Germany
- Tekelec Airtronic GmbH, Muenchen, Germany

TOC

- Jim Tsillas
- Trans-Ameritech Enterprises, Inc., Santa Clara, CA
- Unifix Software GmbH, Braunschweig, Germany
- Vixie Enterprises, La Honda, CA
- Walnut Creek CDROM, Concord, CA
- Xtreme s.a.s., Livorno, Italy

The XFree86 Project, Inc, welcomes the additional contribution of funding and/or equipment. Such contributions should be tax-deductible; we will know for certain when the lawyers get finished with the papers. For more information, contact The XFree86 Project, Inc, at <*BOD@XFree86.org*>

8. Source and binary archive sites

Source patches are available to upgrade X11R6.3 PL2 from the X Consortium (now The Open Group) to XFree86 3.3.3.1. Binaries for many OSs are also available. The distribution is available from:

- ftp://ftp.XFree86.org/pub/XFree86

and the following mirror sites:

- North America:
  - ftp://ftp2.XFree86.org/pub/XFree86 (source and binaries)
  - ftp://ftp.infomagic.com/pub/mirrors/XFree86-current (source and binaries)
  - ftp://ftp.rge.com/pub/X/XFree86 and http://www.rge.com/pub/X/XFree86 (source and binaries)
  - ftp://ftp.varesearch.com/pub/mirrors/xfree86 (source and binaries)
  - ftp://ftp.cs.umn.edu/pub/XFree86 (source and binaries)
  - ftp://ftp.kernel.org/pub/mirrors/xfree86 (source and binaries)
- Europe:
  - ftp://fvkma.tu-graz.ac.at/pub/XFree86 (source and binaries)
  - ftp://gd.tuwien.ac.at/hci/X11/XFree86 and http://gd.tuwien.ac.at/hci/X11/XFree86 (source and binaries)
  - ftp://ftp.fee.vutbr.cz/pub/XFree86 (source patches and binaries)
  - ftp://ftp.gwdg.de/pub/xfree86/XFree86 (source and binaries)
  - ftp://ftp.mpi-sb.mpg.de/pub/X/mirror/ftp.xfree86.org (source and binaries)
  - ftp://ftp.cs.tu-berlin.de/pub/X/XFree86 (source and binaries)
  - ftp://ftp.uni-erlangen.de/pub/Linux/MIRROR.xfree86 (source and Linux binaries)
  - ftp://ftp.uni-stuttgart.de/pub/X11/Xfree86 (source and binaries)
  - ftp://ftp.funet.fi/pub/X11/XFree86 (source and binaries)
  - ftp://ftp.ibp.fr/pub/X11/XFree86 (source and binaries)
  - ftp://ftp.unina.it/pub/XFree86 (source and binaries)
  - ftp://ftp.pvv.unit.no/pub/XFree86 (source and binaries)
  - ftp://sunsite.doc.ic.ac.uk/packages/XFree86 (source and binaries)
- Asia/Australia:
  - ftp://x.physics.usyd.edu.au/pub/XFree86 (source and binaries)
  - ftp://ftp.netlab.is.tsukuba.ac.jp/pub/XFree86 (source and binaries)
  - ftp://ftp.iij.ad.jp/pub/X/XFree86/XFree86 (source and binaries)
  - ftp://ftp.kreonet.re.kr/pub/Linux/xfree86 (source and binaries)

Ensure that you are getting XFree86 3.3.3.1 - some of these sites may archive older releases as well. Check the RELNOTES to find which files you need to take from the archive.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/README.sgml,v 3.75.2.54 1999/08/17
07:39:29 hohndel Exp $
```

TOC

```
$XConsortium: README.sgml /main/31 1996/10/28 05:43:24 kaleb $
```

Configuring XFree86: A Step-By-Step Guide

David Wexelblat and The XFree86 Project, Inc

5 October 1994

This document describes how to set up your XFree86 server and the corresponding `XF86Config` configuration file. If you follow the procedures in this document, you should have no problems getting your server up and running quickly. This document is designed to be generic. Be certain to refer to the operating system specific `README` file for your OS (e.g. `README.SVR4`) and the card/chipset specific `README` file for you video card (e.g. `README.trident`). Where these specific files contradict this generic file, you should follow the specific instructions (there shouldn't be much of that, though).

---

1. Procedure Overview
2. Setting Up The Correct Default Server
3. The Easy Parts of XF86Config
4. Configuring the Video Hardware
5. Configuring the Monitor and its Modes
6. Combining the Video Hardware and Monitor Data
7. Generic Video Modes

1. Procedure Overview

There are two steps to getting things up and running. The first is to select the appropriate server that you will be using and set it up as the default server. The second step is to set up the `XF86Config` file. This file is used to configure the server for your pointer device (e.g. mouse, trackball), video card, and monitor, as well as a few other things.

The `XF86Config` file contains several sections; these procedures will lead you through filling out each part. There is a default/sample `XF86Config` file in `/usr/X11R6/lib/X11/XF86Config.sample`; you should copy this to `/usr/X11R6/lib/X11/XF86Config`, and edit that file to your specific configuration. The *XF86Config(4/5)* manual page describes the `XF86Config` file contents and options in detail. Be sure to read through that manual page as you fill in your `XF86Config` file.

The sections of the `XF86Config` file are:

Files

Sets the default font and RGB paths.

Server Flags

Sets a few general server options. Refer to the manual page to learn about these.

Keyboard

Sets up keyboard devices, and sets a few optional parameters.

Pointer

Sets up the pointer devices, and sets a few optional parameters.

Monitor

Describes your monitor(s) to the server.

Graphics Device

Describes your video hardware to the server.

Screen.

Describes how the monitor and video hardware should be used.

2. Setting Up The Correct Default Server

The default server name is `/usr/X11R6/bin/X`. This is a link to a specific server binary `XF86_xxxx`, located in `/usr/X11R6/bin/`. You should check which server the X link is connected to. If it is not correct, remove it and make a new link to the correct binary. The server binaries are:

XF86_SVGA:

Super-VGA server. Contains accelerated support for Cirrus 542{0,2,4,6,8,9}, 543{0,4} and Western Digital 90C3{1,3} and Oak Technologies Inc. OTI087 chipsets, unaccelerated for the rest of the supported chipsets.

XF86_Mono:

(S)VGA monochrome, optionally Hercules or other monochrome hardware support is linked in.

XF86_VGA16:

Generic VGA 16-color server.

XF86_S3:

S3 accelerated server.

XF86_Mach32:

ATI Mach32 accelerated server.

XF86_Mach64:

ATI Mach64 accelerated server.

XF86_Mach8:

ATI Mach8 accelerated server.

XF86_8514:

8514/A accelerated server.

XF86_P9000:

P9000 accelerated server.

XF86_AGX:

AGX accelerated server.

XF86_W32:

ET4000/W32 and ET6000 accelerated server.

There is a manual page for each of these servers; refer to the manual page for specific details on

supported chipsets and server-specific configuration options.

Note that it is possible to modify the drivers configured into a server via the LinkKit; the server binary may not contain all of the possible drivers, depending on how the distribution was assembled. You can run `/usr/X11R6/bin/X -showconfig` to get a printout of the configured drivers. If you need to relink your server, refer to the `README` file in the LinkKit for specific information.

Configuring XFree86: A Step-By-Step Guide **:** *Setting Up The Correct Default Server*
Previous: *Procedure Overview*
Next: *The Easy Parts of XF86Config*

3. The Easy Parts of XF86Config

The "Files" section of the `XF86Config` file contains the path to the RGB database file (which should, in general, never need to be changed), and the default font path. You can have multiple FontPath lines in your `XF86Config`; they are concatenated. Ensure that each directory listed exists and is a valid font directory. If the server complains about "Can't open default font 'fixed'", it is because there is an invalid entry in your font path. Try running the 'mkfontdir' command in each directory if you are certain that each one is correct. The *XF86Config(4/5)* manual page describes other parameters that may be in this section of the file.

Next comes the "Keyboard" section. In this section, you can specify the keyboard protocol (Xqueue or Normal), the repeat rate, and the default mapping of some of the modifier keys. In general, nothing will need to be modified here. Users of non-English keyboards might want to change the definitions of the modifier keys. See the *XF86Config(4/5)* man page for details.

After this comes the "Pointer" section. In this section you can specify the pointer protocol and device. Note that the protocol name does not always match the manufacturer's name. For example, some Logitech mice (especially newer ones) require either the MouseMan or Microsoft protocols, not the Logitech protocol. Some other mouse parameters can be adjusted here. If you are using a two-button mouse, uncomment the Emulate3Buttons keyword - in this mode, pressing both mouse buttons simultaneously causes the server to report a middle button press.

Note that if the server complains about being unable to open your mouse device, this is NOT a server problem. It has been a very common misconfiguration error on several of the OSs, and 99.999% of the time it is because the device is not correctly configured in the OS. Hence don't bug us until after you prove that your OS level support is correct.

4. Configuring the Video Hardware

The video hardware is described in the "Device" sections. Multiple device sections are permitted, and each section describes a single graphics board.

Be sure to read the server manual pages and the chipset-specific `README` files for any non-generic information that may apply to your setup.

To create a Device section you need to collect the data for your hardware, and make some configuration decisions. The hardware data you need is:

- Chipset
- Amount of video memory
- Dot-clocks available or clock chip used (if programmable)
- Ramdac type (for some servers)

The server, in general, is capable of filling these on its own, but it is best to fully specify things in the `XF86Config` file, so that no mistakes are made. The 'Chipset' is one of the keyword strings for a configured driver (which can be displayed by running 'X -showconfig'). Of the accelerated servers, only some have chipset drivers currently. The amount of memory is specified in KBytes, so 1M of memory would be specified as 1024.

The dot-clocks are the trickiest part of card configuration. Fortunately a large database of collected dot-clocks is available. A list of Device entries for some graphics boards can be found in the `Devices' file. If you find one for your card, you can start with that. Also, the first part of the `modeDB.txt` file lists information for a myriad of SVGA cards. For accelerated cards, you can also look in the `AccelCards' file. If you are fortunate, your card is listed in one place or the other. If you find your card, copy the numbers from the database to the Clocks line in your `XF86Config` file, exactly as they appear in the database, without sorting, and leaving any duplicates. Note that some of the newer accelerated cards use a programmable clock generator, in which case a ClockChip line is used in your `XF86Config` file to identify the type of clock generator. (e.g. 'ClockChip "icd2061a"', which would be used for a #9 GXe board).

If you can't find a listing for your board, you can attempt to have the server detect them. Run the command 'X -probeonly >/tmp/out 2>&1' (for sh or ksh) or 'X -probeonly >&/tmp/out' (for csh). Be sure that the `XF86Config` file does **not** contain a Clocks line at this point. Running this will cause your monitor to freak out for a couple of seconds, as the server cycles through the clocks rapidly. It should not damage your monitor, but some newer monitors may shut themselves off because things may go out of spec. Anyhow, when this gets done, look in the file /tmp/out for the detected dot-clocks. Copy these to the Clocks line in your `XF86Config` file, exactly as they appear in /tmp/out. Don't sort them or rearrange them in any way.

It is possible that your board has a programmable clock generator. A symptom of this will be a printout of only 2 or 3 clock values, with the rest all zeros. If you run into this, and your board is not listed in the databases, contact the XFree86 team for help, or post a message to comp.windows.x.i386unix. Note that most current Diamond hardware falls into this category, and Diamond will not release the programming details, so we can't help you. There are some ethically questionable solutions available that you can inquire about on netnews; we do not advocate these methods, so do not contact us about them.

Some servers (S3 and AGX) require you to identify the type and speed of the RAMDAC your board uses in order to get the most out of the hardware. This is done by adding 'Ramdac' and 'DacSpeec' entries. For details of the supported RAMDACs, refer to the appropriate server manual page. Note, in previous versions of XFree86 the RAMDAC type was specified with an Option flag.

You may need to specify some Option flags for your hardware The server manual pages will describe these options, and the chipset-specific `README` files will tell you if any are required for your board.

Configuring XFree86: A Step-By-Step Guide **:** *Configuring the Video Hardware*
Previous: *The Easy Parts of XF86Config*
Next: *Configuring the Monitor and its Modes*

5. Configuring the Monitor and its Modes

Configuring monitor modes can be a trying experience, unfortunately, because of the lack of standardization in monitor hardware. We have attempted to simplify this by collecting databases of specific monitor information, and assembling a set of "generic" modes that should get pretty much any monitor up and functional. For all the gory details of mode generation and tuning, refer to the `VideoModes.doc' document by Eric Raymond.

The monitor specs and video modes are described in the "Monitor" sections in the `XF86Config` file. To create a Monitor section, you need to know your monitor's specifications. In particular, you need to know what range of horizontal sync and vertical sync (refresh) rates it supports and what its video bandwidth is. This information should be available in the monitor's user manual. Also check the 'Monitors' file to see if it has an entry for your monitor. See the *XF86Config(4/5)* manual page for details of how this information is entered into the Monitor section.

Next, you need to provide a set of video modes that are suitable for the monitor. The first step is to check in the 'Monitors' and `modeDB.txt` files to see if there is a listing of modes for your specific monitor. If there is, copy those modes to the Monitor section of your `XF86Config` file. Verify that there is a clock listed on the Clocks line in your `XF86Config` that matches the dot-clock in the 2nd parameter of each mode line; delete any mode line that does not have a matching clock on your card. If you still have modes left, you are in good shape.

If you don't find any specific modes, or need more modes for the resolutions you want to use, refer to the Generic Video Modes listing below. Match the mode specification against your monitor's specifications; pick the highest-refresh mode that is within specs, and make sure you have a matching dot-clock on your Clocks line. Try the VESA modes before any corresponding alternate mode setting. Copy the mode specification to the Monitor section of your `XF86Config` file. Note that these modes are likely not optimal; they may not be sized perfectly, or may not be correctly centered. But they should get you up and running. If you want to tune the mode to your monitor, you can read the 'Fixing Problems with the Image' section of the VideoModes.doc file.

A note before you are done. If the same mode name occurs more than once in the Monitor section of the `XF86Config` file, the server will use the first mode with a matching clock. It is generally considered a bad idea to have more than one mode with the same name in your `XF86Config` file.

6. Combining the Video Hardware and Monitor Data

Once you have given a description of your monitor and graphics hardware you need to specify how they are to be used by the servers. This is done with the "Screen" sections in the `XF86Config` file. You need to supply a Screen section for each of the server driver types you will be using. The driver types are "SVGA" (XF86_SVGA), "VGA16" (XF86_VGA16), "VGA2" (XF86_Mono), "MONO" (XF86_Mono, XF86_VGA16), and "ACCEL" (XF86_S3, XF86_Mach32, XF86_Mach8, XF86_Mach64, XF86_8514, XF86_P9000, XF86_AGX, XF86_W32). Each Screen section specifies which Monitor description and Device description are to be used.

The Screen sections include one or more "Display" subsections. One Display subsection may be provided for each depth that the server supports. In the Display subsection you can specify the size of the virtual screen the server will use. The virtual screen allows you to have a "root window" larger than can be displayed on your monitor (e.g. you can have an 800x600 display, but a 1280x1024 virtual size). The Virtual keyword is used to specify this size. Note that many of the new accelerated server use non-displayed memory for caching. It is not desirable to use all of your memory for virtual display, as this leaves none for caching, and this can cost as much as 30-40% of your server performance.

The last thing you specify in Display subsection is the display modes. These are the physical display resolutions that the server will use. The name is arbitrary, but must match something in the appropriate Monitor section. In general, these names are the display resolution (e.g. "1024x768"), but need not be. You can list as many as desired; the first is the default/starting display, and you can cycle through the list with Ctrl-Alt-Keypad+ or Ctrl-Alt-Keypad- hotkey sequences.

That's it. Now you're ready to test out your new XFree86 installation.

7. Generic Video Modes

```
#
#  Mode       Refresh  Hor. Sync  Dot-clock  Interlaced?  VESA?
#  ---------------------------------------------------------------
#  640x480     60Hz      31.5k     25.175M       No         No
#  640x480     60Hz      31.5k     25.175M       No         No
#  640x480     63Hz      32.8k     28.322M       No         No
#  640x480     70Hz      36.5k     31.5M         No         No
#  640x480     72Hz      37.9k     31.5M         No        Yes
#  800x600     56Hz      35.1k     36.0M         No        Yes
#  800x600     56Hz      35.4k     36.0M         No         No
#  800x600     60Hz      37.9k     40.0M         No        Yes
#  800x600     60Hz      37.9k     40.0M         No         No
#  800x600     72Hz      48.0k     50.0M         No        Yes
#  1024x768i   43.5Hz    35.5k     44.9M        Yes         No
#  1024x768    60Hz      48.4k     65.0M         No        Yes
#  1024x768    60Hz      48.4k     62.0M         No         No
#  1024x768    70Hz      56.5k     75.0M         No        Yes
#  1024x768    70Hz      56.25k    72.0M         No         No
#  1024x768    76Hz      62.5k     85.0M         No         No
#  1280x1024i  44Hz      51kHz     80.0M        Yes         No
#  1280x1024i  44Hz      47.6k     75.0M        Yes         No
#  1280x1024   59Hz      63.6k    110.0M         No         No
#  1280x1024   61Hz      64.24k   110.0M         No         No
#  1280x1024   74Hz      78.85k   135.0M         No         No

#
# 640x480@60Hz Non-Interlaced mode
# Horizontal Sync = 31.5kHz
# Timing: H=(0.95us, 3.81us, 1.59us), V=(0.35ms, 0.064ms, 1.02ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "640x480"     25.175  640  664  760  800    480  491  493  525

#
# Alternate 640x480@60Hz Non-Interlaced mode
# Horizontal Sync = 31.5kHz
# Timing: H=(1.27us, 3.81us, 1.27us) V=(0.32ms, 0.06ms, 1.05ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "640x480"     25.175  640  672  768  800    480  490  492  525

#
# 640x480@63Hz Non-Interlaced mode (non-standard)
# Horizontal Sync = 32.8kHz
# Timing: H=(1.41us, 1.41us, 5.08us) V=(0.24ms, 0.092ms, 0.92ms)
```


```
#
# name        clock   horizontal timing     vertical timing      flags
 "640x480"     28.322  640  680  720  864    480  488  491  521

#
# 640x480@70Hz Non-Interlaced mode (non-standard)
# Horizontal Sync = 36.5kHz
# Timing: H=(1.27us, 1.27us, 4.57us) V=(0.22ms, 0.082ms, 0.82ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "640x480"     31.5    640  680  720  864    480  488  491  521

#
# VESA 640x480@72Hz Non-Interlaced mode
# Horizontal Sync = 37.9kHz
# Timing: H=(0.76us, 1.27us, 4.06us) V=(0.24ms, 0.079ms, 0.74ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "640x480"     31.5    640  664  704  832    480  489  492  520

#
# VESA 800x600@56Hz Non-Interlaced mode
# Horizontal Sync = 35.1kHz
# Timing: H=(0.67us, 2.00us, 3.56us) V=(0.03ms, 0.063ms, 0.70ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "800x600"     36      800  824  896 1024    600  601  603  625

#
# Alternate 800x600@56Hz Non-Interlaced mode
# Horizontal Sync = 35.4kHz
# Timing: H=(0.89us, 4.00us, 1.11us) V=(0.11ms, 0.057ms, 0.79ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "800x600"     36      800  832  976 1016    600  604  606  634

#
# VESA 800x600@60Hz Non-Interlaced mode
# Horizontal Sync = 37.9kHz
# Timing: H=(1.00us, 3.20us, 2.20us) V=(0.03ms, 0.106ms, 0.61ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "800x600"     40      800  840  968 1056    600  601  605  628 +hsync +vsync

#
# Alternate 800x600@60Hz Non-Interlaced mode
# Horizontal Sync = 37.9kHz
# Timing: H=(1.20us, 3.80us, 1.40us) V=(0.13ms, 0.053ms, 0.69ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "800x600"     40      800 848 1000 1056     600  605  607  633

#
# VESA 800x600@72Hz Non-Interlaced mode
```


```
# Horizontal Sync = 48kHz
# Timing: H=(1.12us, 2.40us, 1.28us) V=(0.77ms, 0.13ms, 0.48ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "800x600"     50      800  856  976 1040    600  637  643  666  +hsync +vsync

#
# 1024x768@43.5Hz, Interlaced mode (8514/A standard)
# Horizontal Sync = 35.5kHz
# Timing: H=(0.54us, 1.34us, 1.25us) V=(0.23ms, 0.23ms, 0.93ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1024x768i"   44.9   1024 1048 1208 1264    768  776  784  817  Interlace

#
# VESA 1024x768@60Hz Non-Interlaced mode
# Horizontal Sync = 48.4kHz
# Timing: H=(0.12us, 2.22us, 2.58us) V=(0.06ms, 0.12ms, 0.60ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1024x768"    65     1024 1032 1176 1344    768  771  777  806 -hsync -vsync

#
# 1024x768@60Hz Non-Interlaced mode (non-standard dot-clock)
# Horizontal Sync = 48.4kHz
# Timing: H=(0.65us, 2.84us, 0.65us) V=(0.12ms, 0.041ms, 0.66ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1024x768"    62     1024 1064 1240 1280   768  774  776  808

#
# VESA 1024x768@70Hz Non-Interlaced mode
# Horizontal Sync=56.5kHz
# Timing: H=(0.32us, 1.81us, 1.92us) V=(0.05ms, 0.14ms, 0.51ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1024x768"    75     1024 1048 1184 1328    768  771  777  806 -hsync -vsync

#
# 1024x768@70Hz Non-Interlaced mode (non-standard dot-clock)
# Horizontal Sync=56.25kHz
# Timing: H=(0.44us, 1.89us, 1.22us) V=(0.036ms, 0.11ms, 0.53ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1024x768"    72     1024 1056 1192 1280    768  770  776 806   -hsync -vsync

#
# 1024x768@76Hz Non-Interlaced mode
# Horizontal Sync=62.5kHz
# Timing: H=(0.09us, 1.41us, 2.45us) V=(0.09ms, 0.048ms, 0.62ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1024x768"    85     1024 1032 1152 1360    768  784  787  823
```


```
#
# 1280x1024@44Hz, Interlaced mode
# Horizontal Sync=51kHz
# Timing: H=(0.02us, 2.7us, 0.70us) V=(0.02ms, 0.24ms, 2.51ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1280x1024i"  80     1280 1296 1512 1568   1024 1025 1037 1165  Interlace

#
# Alternate 1280x1024@44Hz, Interlaced mode (non-standard dot-clock)
# Horizontal Sync=47.6kHz
# Timing: H=(0.42us, 2.88us, 0.64us) V=(0.08ms, 0.12ms, 0.96ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1280x1024i"  75     1280 1312 1528 1576   1024 1028 1034 1080  Interlace

#
# 1280x1024@59Hz Non-Interlaced mode (non-standard)
# Horizontal Sync=63.6kHz
# Timing: H=(0.36us, 1.45us, 2.25us) V=(0.08ms, 0.11ms, 0.65ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1280x1024"  110     1280 1320 1480 1728   1024 1029 1036 1077

#
# 1280x1024@61Hz, Non-Interlaced mode
# Horizontal Sync=64.25kHz
# Timing: H=(0.44us, 1.67us, 1.82us) V=(0.02ms, 0.05ms, 0.41ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1280x1024"  110     1280 1328 1512 1712   1024 1025 1028 1054

#
# 1280x1024@74Hz, Non-Interlaced mode
# Horizontal Sync=78.85kHz
# Timing: H=(0.24us, 1.07us, 1.90us) V=(0.04ms, 0.04ms, 0.43ms)
#
# name        clock   horizontal timing     vertical timing      flags
 "1280x1024"  135     1280 1312 1456 1712   1024 1027 1030 1064
```

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Config.sgml,v 3.11.2.1 1998/04/29
04:18:28 dawes Exp $
```

```
$XConsortium: Config.sgml /main/7 1996/10/19 18:03:03 kaleb $
```

Information for DEC 21030 Users (aka TGA)

TOC

The XFree86 Project, Inc.

23th October 1998

1. DEC 21030

2. Additional Notes

1. DEC 21030

- The DEC 21030 is supported by XFree86 in this release of XFree86 3.3.
- Current Known Problems
  1. Only one modeline is accepted, this will be the first viable one that matches other criteria.
  2. Due to the above, Virtual Resolutions is not supported either.
- The following options may be specified for the 21030 driver:

  Option "dac_8_bit"

  Turn on 8Bit BT485 RamDac (Multia and 8-plane TGA only).

  Option "dac_6_bit"

  Turn on 6Bit BT485 RamDac (Multia and 8-plane TGA only).

  MemBase 0x???????

  If the server does not detect the base address of the 21030, then Check /proc/pci for the 21030 and look for the "Prefetchable 32 bit memory at 0x???????" and enter this as your MemBase setting. In XFree86 v3.3.2, if you are using Linux > v2.0.27 with the PCI routines the server should detect the base address automatically.
- No acceleration features of the 21030 have been taken advantage of yet!

2. Additional Notes

This code has been tested only under Linux on DEC's UDB box (Multia), the ZLZp-E1 (8-plane TGA), and the ZLXp-E2 (24-plane TGA). The ZLXp-E3 (24-plane+3D TGA) is untested but should work.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/DECtga.sgml,v 3.6.2.5 1998/11/07
13:37:46 dawes Exp $
```

Information for Matrox Users

The XFree86 Project Inc.

30 December 1998

1. Supported hardware

    1.1. What's not supported

2. Features:

3. Configuration:

4. Known solutions for some problems:

5. Authors

1. Supported hardware

The current MGA driver in the SVGA server supports

- Matrox Millennium (MGA2064W with Texas Instruments TVP3026 RAMDAC). It has been tested with 175, 220MHz, and 250MHz cards with 2MB, 4MB and 8MB WRAM.
- Millennium II both PCI and AGP (MGA2164W with Texas Instruments TVP3026 RAMDAC). It has been tested with 4 MB, 8 MB and 16 MB WRAM.
- Matrox Mystique (Both MGA1064SG and MGA1164SG with integrated RAMDACs) 170 MHz and 220 MHz (Mystique 220) versions should work.
- Millennium G200 with SGRAM and SDRAM (Millennium G200-SD), with 8MB RAM.
- Mystique G200 (but no TVout support)
- Productiva G100 with SGRAM and SDRAM. 4MB and 8MB versions have been tested.
- Matrox G400 (only the first head and no TVout support).

1.1. What's not supported

- Chipsets other than those listed above. We are still interested in providing support for the other Matrox chipsets including the Impression, Atlas, Genesis etc... but at this time have not been able to obtain documentation for them.
- MGA2064W and MGA2164W based cards with ramdacs other than the TVP3026 RAMDAC (like the Matrox Corona) are not supported.

2. Features:

- uses linear frame buffer
- Resolutions up to the maximum supported by the card should be possible.
- 8 bpp, 16 bpp (depth 15 and 16), 24 bpp (depth 24, packed) and 32 bpp (depth 24, sparse) are all supported.
- supports VESA Display Power Management Signaling (DPMS)
- supports RGB Sync-on-Green
- supports the XF86_DGA extension
- Makes extensive use of the graphics accelerator. This server is very well accelerated, and is one of the fastest XFree86 X servers.

3. Configuration:

The MGA driver should auto-detect all supported hardware so you needn't have anything other than the Identifier in the Section "Device" of the XF86Config file. When running the XF86Setup or xf86config programs one merely needs to select a Matrox card so that the correct server will be used. One need not and should not specify a RAMDAC, clockchip or allow the setup program to probe for clocks. The driver will auto-detect the amount of video ram present, however, due to some hardware problems this is not detected for the MGA2164W (Millennium II) or G100/G200. In this case users should specify the amount of video ram in the Section "Device" of the XF86Config file. eg:

```
VideoRam 4096
    or
VideoRam 8192
    or
VideoRam 16384
```

as appropriate so that the server doesn't have to probe for it.

The following Section "Device" options are supported by the MGA driver:

- Option "sw_cursor"

  Will disable the hardware cursor on the Millennium and Millennium II.
- Option "no_accel"

  Will disable all hardware acceleration (oh my!).
- Option "no_pixmap_cache"

  Will disable caching of pixmaps in offscreen video memory.
- Option "sync_on_green"

  Will enable syncing on green for sync-on-green monitors (these are typically fixed frequency workstation monitors).
- Option "pci_retry"

  This will allow the MGA hardware to generate a pci_disconnect based on accelerator FIFO status. This can yield large performance boosts for some graphics operations but has a tendency to hog the PCI bus so it is turned off by default.
- Option "mga_sdram"

  This will force the server to disable sgram features such as block mode fills and hardware planemasks.

TOC

4. Known solutions for some problems:

- Temporary loss of monitor sync when the cursor shape changes on Millennium and Millennium II. The hardware cursor has been enabled by default in 3.3.3.1. This seems to cause some problems on a minority of systems. If you experience problems with this on your system, please put:

  ```
  Option "sw_cursor"
  ```

  in the Section "Device" of the XF86Config file to disable the hardware cursor.
- Garbage in the cursor instead of the normal cursor image. A bug in the driver will cause this when less than 1K of video memory is left unused and the hardware cursor is enabled for some cards. If you experience this problem, please put:

  ```
  Option "sw_cursor"
  ```

  in the Section "Device" of the XF86Config file to disable the hardware cursor. This should be fixed in XFree86 3.3.3.1 as in cases like this the software cursor should be used automatically.
- the driver doesn't support some values of HTotal parameter in Modelines in the XF86Config file. If you get flickering vertical stripes on the screen, try to change this parameter +/- 8.
- On some Millennium II cards the driver shows severe distortions with 24bpp in modes above about 1024x768. We hope to have automated the detection and fix of this problem. If it still occurs, putting

  ```
  Option "mga_24bpp_fix"
  ```

  in the Device Section may fix the problem.
- On some MGA cards the amount of memory is mis-detected, on others probing for the amount of memory can cause a lockup in the system so memory probing is not done on those hardware (Millennium II, G100/G200). If the default of 4MB RAM (Millennium II) or 8MB RAM (G100/G200) is not correct, specify the amount of video ram in the Section "Device" of the XF86Config file as described in section 3 above.
- If you Millennium II card that worked fine with XFree86-3.3.2.3 and earlier now shows pixel errors and strange effects when returning to the text console, make sure that the amount of memory that the server reports is correct. See item above for details.
- With virtual screens that use 8MB of memory or more (e.g., 2048x2048 at 16bpp) there can be cursor distortions when panning the screen vertically. If that occurs, please put

  ```
  Option "sw_cursor"
  ```

  in the Section "Device" of the XF86Config file to disable the hardware cursor.


TOC

TOC

5. Authors

Radoslaw Kapitan, *kapitan@student.uci.agh.edu.pl*

Mark Vojkovich, *mvojkovi@sdcc10.ucsd.edu*

and:

- Andrew Vanderstock, *vanderaj@mail2.svhm.org.au*
- Angsar Hockmann, *Ansgar.Hockmann@hrz.uni-dortmund.de*
- Michael Will, *Michael.Will@student.uni-tuebingen.de*
- Andrew Mileski, *aem@ott.hookup.net*
- Stephen Pitts, *pitts2@memphisonline.com*
- Dirk Hohndel, *hohndel@XFree86.Org*
- Leonard N. Zubkoff, *lnz@dandelion.com*
- Harm Hanemaayer, *H.Hanemaayer@inter.nl.net*
- Guy Desbief, *g.desbief@aix.pacwan.net*
- Takaaki Nomura, *tnomura@sfc.keio.ac.jp*
- Doug Merritt, *doug@netcom.com*

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/MGA.sgml,v 3.4.2.16 1999/06/23
12:37:18 hohndel Exp $
```

Notes for Mach32 X Server

Bryan Feir (jenora@istar.ca)

2 July 1997

1. Supported Cards, RAMDACs, and Bits Per Pixel

2. XF86Config options

3. Known Problems and Bug Reports

1. Supported Cards, RAMDACs, and Bits Per Pixel

The base support in the Mach32 X server is for 8 bpp, with a dot clock of up to 80 MHz. At present 15/16 bpp is supported on only three of the many RAMDACs; however those three cover the most commonly sold cards.

RAMDAC	Max Dot Clock	BPP	Max Resolution	Video RAM Required
Default	80MHz	8	1280x1024i	2Mb
Default	80MHz	8	1024x768	1Mb
ATI68875	135MHz	8	1280x1024	2Mb
ATI68875	80MHz	16	1024x768	2Mb
AT&T20C49x	80MHz	8	1024x768	1Mb
AT&T20C49x	40MHz	16	800x600	2Mb
BT481	80MHz	8	1024x768	1Mb
BT481	40MHz	16	800x600	2Mb

The RAMDAC is reported when you run the Mach32 X server with the "`-probeonly`" command line option, or can be specified in the XF86Config file.

The ATI68875 (or the TLC34075) is used on the Graphics Ultra + and the Graphics Ultra Pro. The Brooktree 481 is used on most Graphics Wonder cards. The AT&T20C491 is used on many of the OEM cards that are built into component systems.

The BIOS detection unfortunately lumps the BT481 and the AT&T20C49x together, while they require different configuration controls in 16 bit mode. SuperProbe can tell the difference, and will report which it finds. In the server itself, the BT481 is considered the default value. If you have an AT&TC49x RAMDAC on your card you will have to include the Ramdac entry in the XF86Config file as below.

2. XF86Config options

Several options are supported in the "Device" section for the Mach32 X server. Most of them should be auto-detected if needed, but a few may need to be deliberately set. For example, the "Clocks" entry should almost certainly be set after first running the server with the `-probeonly` option, so as to avoid the probe in later runs.

Option "composite"

This option will set the composite sync for monitors that require this.

Option "dac_8_bit"

This option enables 8 bits per RGB value. Note this option does not work with all RAMDACs, and is not considered supported by the Mach32 itself.

Option "ast_mach32"

This option sets some special handling for the AST version of the Mach32 card that comes soldered in to some of their motherboards. This card will lock up without this option.

Option "intel_gx"

This option sets the memory aperture address to the hardwired value for the Intel GX Pro. It is equivalent to setting Membase to 0x78000000.

Option "no_linear"

This option disables the use of the linear mapped framebuffer. This should be auto-detected.

Option "sw_cursor"

This option allows you to use the software cursor instead of the hardware cursor.

MemBase baseaddress

This entry specifies the video memory aperture address. Normally the aperture address is automatically determined, but on some VESA Local Bus systems the address chosen will not work. If the Mach32 X server is dying with a seg. fault, then try setting the aperture address to another location.

Clocks *clock* ...

This entry gives the clock rates for the server to use.

Ramdac "*type*"

This entry specifies the RAMDAC type. The following values are valid for *type*:

- ati68830


TOC

- sc11483
- sc11486
- sc11488
- ims_g173
- mu9c4870
- ati68875*
- bt885
- tlc34075*
- bt476
- bt478
- inmos176
- inmos178
- bt481*
- bt482
- ims_g174
- mu9c1880
- mu9c4910
- sc15025
- sc15026
- att20c490*
- ati68860
- stg1700
- sc15021
- stg1702
- att21c498

Only the ones marked with [*] have an effect yet.

Notes for Mach32 X Server **:** *XF86Config options*
Previous: *Supported Cards, RAMDACs, and Bits Per Pixel*
Next: *Known Problems and Bug Reports*

3. Known Problems and Bug Reports

There are several known problems with the current version of the Mach32 X server. They include:

- Not all RAMDACs are supported at higher colour ranges, and not all that are can be detected properly. In fact most of the RAMDAC values above have no effect except to block higher bit modes.
- Sixteen bit character support (e.g., Chinese and Japanese character sets) has been known to lose parts of characters. While this should be fixed, if it occurs try running the server with Option "no_linear".

Bug reports should be sent to XFree86@XFree86.org or posted to the comp.windows.x.i386unix newsgroup.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Mach32.sgml,v 3.5.2.3 1998/11/07
13:37:47 dawes Exp $
```

```
$XConsortium: Mach32.sgml /main/4 1996/10/28 04:47:43 kaleb $
```

Mach64 X Server Release Notes

Kevin E. Martin (martin@cs.unc.edu)

1999 June 28

1. Supported Cards, RAMDACs, and Bits Per Pixel

2. Optimizing the speed of the Mach64 X server

3. XF86Config options

4. Enhancements for this release

5. Cards known to work with this release

6. Known Problems and Bug Reports

1. Supported Cards, RAMDACs, and Bits Per Pixel

The Mach64 X server supports 8bpp with a dot clock up to 80MHz on all Mach64 based cards. On most cards, higher dot clocks and additional depths are available (see the table below). What determines this support is the RAMDAC on your card.

RAMDAC	Max Dot Clock	BPP	Max Resolution	Video RAM Required
ATI68860	135MHz	8	1280x1024	2Mb
ATI68860	135MHz	16	1280x1024	4Mb
ATI68860	80MHz	32	1024x768	4Mb
ATI68875	80MHz	32	1024x768	4Mb
CH8398	135MHz	8	1280x1024	2Mb
CH8398	80MHz	16	1024x768	2Mb
CH8398	40MHz	32	800x600	2Mb
STG1702	135MHz	8	1280x1024	2Mb
STG1702	80MHz	16	1024x768	2Mb
STG1702	50MHz	32	800x600	2Mb
STG1703	135MHz	8	1280x1024	2Mb
STG1703	80MHz	16	1024x768	2Mb
STG1703	50MHz	32	800x600	2Mb
AT&T20C408	135MHz	8	1280x1024	2Mb
AT&T20C408	80MHz	16	1024x768	2Mb
AT&T20C408	40MHz	32	800x600	2Mb
3D Rage II	170MHz	8	1600x1200	4Mb
3D Rage II	170MHz	16	1600x1200	4Mb
3D Rage II	170MHz	32	1024x768	4Mb
3D Rage II+DVD	200MHz	8	1600x1200	4Mb
3D Rage II+DVD	200MHz	16	1600x1200	4Mb
3D Rage II+DVD	200MHz	32	1024x768	4Mb
Rage Pro	230MHz	8	1600x1200	8Mb
Rage Pro	230MHz	16	1600x1200	8Mb

TOC

Rage Pro	230MHz	32	1600x1200	8Mb
Internal	135MHz	8	1280x1024	2Mb
Internal	80MHz	16	1024x768	2Mb
Internal	40MHz	32	800x600	2Mb
IBM RGB514	220MHz	8	1600x1200	2Mb
IBM RGB514	220MHz	16	1600x1200	4Mb
IBM RGB514	135MHz	32	1024x768	4Mb
All Others[*]	80MHz	8	1280x1024	2Mb

[*] - The dot clocks are limited to 80MHz and the bpp is limited to 8.

The table above specifies the maximum resolution and the video memory required to run this maximum resolution. Smaller resolutions will require less video memory.

The RAMDAC is reported when you run the Mach64 X server with the "`-probeonly`" command line option. The RAMDAC reported should be correct for all Mach64 cards. It can also be specified in the XF86Config file, but this is not recommended unless the RAMDAC reported in the probeonly output is incorrect. Before specifying the RAMDAC in your XF86Config file visually verify which RAMDAC is on your Mach64 card. If the RAMDAC reported in the probeonly output is definitely different than what you see on the card, then check to see if you have a RAMDAC specified in your XF86Config file. If you do, comment this line out and re-run the Mach64 X server with the "`-probeonly`". If it still reports the incorrect RAMDAC, please send in a bug report to XFree86@XFree86.Org.

The ATI68860 RAMDACs are usually found on ATI Graphics Pro Turbo and ATI WinTurbo cards. The IBM RGB514 RAMDAC is found on the ATI Graphics Pro Turbo 1600 card. The other RAMDACs are usually found on ATI Graphics Xpression, ATI Video Xpression and ATI 3d Xpression cards. Mach64 CT, ET, VT, VT3, VT4, LT, GT (3D Rage), 3D Rage II, 3D Rage IIC, 3D Rage II+DVD, Rage Pro, and Rage LT Pro chips have an "Internal" RAMDAC (i.e., it is built into the Mach64 chip).

As advertised, Mach64 graphics cards can use a special 24bpp mode (packed pixel mode), but this is not currently supported in the Mach64 X server. This will be added in the next major release.

The Mach64 X server requires the video memory aperture to function properly. This means that ISA Mach64 cards in systems with more than 12Mb of main memory will not work. If you have a PCI based Mach64 card or a VLB based Mach64 card, then the Mach64 X server will work with any amount of main memory.

Accelerated doublescan modes are supported on VT, VT3, VT4, LT, GT, Rage II, Rage IIC, Rage II+DVD, Rage Pro and Rage LT Pro based Mach64 cards. Mach64 cards with other chips cannot handle accelerated double scan modes due to a hardware limitation. Non-accelerated doublescan modes should work with the ATI driver in the SVGA X server for all Mach64 cards.

2. Optimizing the speed of the Mach64 X server

To maximize the speed of the Mach64 X server, I suggest that you use the following maximum resolutions. This will allow room for the font and pixmap caches and a hardware cursor.

Max Resolution	BPP	Video RAM
1600x1200	8	8Mb
1600x1200	16	8Mb
1280x1024	32	8Mb
1280x1024	8	4Mb
1280x1024	16	4Mb
1024x767	32	4Mb
1280x1024	8	2Mb
1024x767	16	2Mb
800x600[*]	32	2Mb
1024x767	8	1Mb
800x600[*]	16	1Mb

[*] - With a 2MB video card, the only way to use the font and pixmap caches is to have a virtual resolution of 1024x480 with a 640x480 mode. I suggest using 800x600 to maximize your screen size at the cost of the speed gained from the caches. The same argument can be made for 1MB video cards running in 16bpp mode. Note that it is not possible to run in 32bpp mode with 1MB of video memory.

Technical explanation for the above suggestions: The Mach64 X server uses a font and pixmap cache that is only available at a screen width of 1024 or greater. This restriction will be removed in a future version of the X server. To obtain the best performance from your video card, you need to make sure that there is enough room off-screen for the caches (at least 1024x256). In addition to the cache, the Mach64 uses memory mapped registers which are mapped to the last 1024 bytes of the memory aperture. This takes away another line from video memory. Thus, you need at least a video memory area of 1024x257.

3. XF86Config options

Several options are supported in the "Device" section for the Mach64 X server. By default, the Mach64 X server will determine the RAMDAC type from the BIOS. If you wish to override the default RAMDAC type (not recommended unless the BIOS incorrectly reports your RAMDAC type), you can specify the RAMDAC type in the XF86Config file with the "Ramdac" entry. The Mach64 X server will also program the clocks based on the clock chip read from the BIOS. If you wish to override the default clock chip type (not recommended unless the BIOS incorrectly reports your clock chip type), you may specify the clock chip in the XF86Config file with the "ClockChip" entry. If, however, you wish to use the preprogrammed clocks, you can turn off the clock programming with the "no_program_clocks" option. In this case, the Mach64 X server reads the Clocks from the BIOS. The "Clocks" lines in the XF86Config file are normally ignored by the Mach64 X server unless the "no_bios_clocks" option is given. Note on newer Mach64 cards (CT, ET, VT, GT, 3D Rage II, 3D Rage II+DVD and Rage Pro) the "Ramdac", "ClockChip" and "Clocks" lines have no meaning and should not be included in your XF86Config file.

Option "sw_cursor"

This option allows you to use the software cursor instead of the hardware cursor.

Option "hw_cursor"

This option turns on the hardware cursor. This should not be necessary since the hardware cursor is used by default unless the "sw_cursor" option is specified.

Option "composite"

This option will set the composite sync for monitors that require this.

Option "dac_8_bit"

This option enables 8 bits per RGB value. Note that this does not work with the Chrontel 8398 RAMDAC. This options is not necessary since 8 bits per RGB value is the default for the Mach64 X server for all Mach64 cards except those with the Chrontel 8398 RAMDAC.

Option "dac_6_bit"

This option enables 6 bits per RGB value.

Option "override_bios"

This option allows you to specify a video mode that the video card's BIOS believes to be illegal. Some BIOSs have incorrect maximum resolution and/or dot clock limitations. Use this option with extreme care. It is possible to specify a video mode that can damage your card or monitor.

Option "no_block_write"

This option allows you to turn off block write mode. Block write mode only works on certain

types of VRAM cards. This option has no effect on DRAM based cards. If you see noise on the screen that can be captured via xmag, then it is probably a problem with block write mode being turned on when it should not. This ``noise" usually looks like bits of windows/menus repeated on the screen.

Option "block_write"

This option allows you to turn on block write mode. Block write mode only works on certain types of VRAM cards, and this option has no effect on DRAM based cards. If you want to override the probed default, you can use this option. Note that this may result in ``noise" appearing on the screen.

Option "power_saver"

This option allows the server to use the power saving features of certain "green" monitors instead of blanking when the screen saver is activated. This option is still experimental.

Option "no_program_clocks"

This option allows you to disable the clock programming. Normally the Mach64 server will program the clocks based on the clock chip type unless this option is given. With this option, the clocks are either read from the BIOS or, if the "no_bios_clocks" option is set, set from the Clocks line.

Option "no_bios_clocks"

This option allows you to override the clocks read from the video card's BIOS and use the clocks specified in the Clocks line in your XF86Config file. Normally the Mach64 server will ignore both the BIOS clocks and the clocks specified in the Clocks line unless the "no_program_clocks" options is set (see above).

Option "no_font_cache"

This option allows you to disable the font cache. By default the font cache is turned on if the horizontal resolution is 1024 pixels or greater and there is enough off-screen video memory to hold the cache.

Option "no_pixmap_cache"

This option allows you to disable the pixmap cache. By default the pixmap cache is turned on if the horizontal resolution is 1024 pixels or greater and there is enough off-screen video memory to hold the cache.

Option "fifo_conservative"

This option allows you to use a more conservative display fifo value. If you are experiencing snow or vertical banding on the screen, try adding this option to see if it fixes the problem.

MemBase baseaddress

This entry specifies the video memory aperture address. By default the aperture address is automatically determined and this option should not be necessary. If the Mach64 X server is dying with a seg. fault, then the memory aperture might not be correctly determined. To fix this try setting the aperture address to another location.

ClockChip "*type*"

This entry specifies the clock chip type. The following values are valid for *type*:

- ati18818
- att20c408
- ch8398
- ibm_rgb514
- ics2595
- stg1703

Ramdac "*type*"

This entry specifies the RAMDAC type. The following values are valid for *type*:

- ati68860
- ati68860b
- ati68860c
- ati68875
- att20c408
- ch8398
- ibm_rgb514
- internal
- stg1702
- stg1703
- tlc34075

DacSpeed "*MHz*"

This entry allows you to override the default maximum dot clock. Use this option with extreme caution. If you specify a *MHz* value too large for your card, you can damage it.

4. Enhancements for this release

With this release, the following enhancements have been made:

- Proper identification of all current Mach64 chips
- Support for VT4 and Rage IIC based cards
- Improved timing calculation for video FIFOs
- Fixed timing bug in font code
- Fixed VGA font restoration bug when exiting the X server

5. Cards known to work with this release

The following is a list of cards that have been tested with this release. Many other cards should work including All-In-Wonder and All-In-Wonder Pro cards as well as motherboards with Mach64, 3D Rage II and Rage Pro included on them. If you have a new card that does not appear to work, see the Known Problems and Bug Reports section below.

ATI Xpert@Play 98	4MB	3D Rage Pro	(AGP)
ATI Xpert 98	4MB	3D Rage Pro	(PCI)
ATI Xpert XL	4MB	3D Rage Pro	(AGP)
ATI Rage IIC	4MB	3D Rage IIC	(AGP)
ATI Xpert@Play	8MB	3D Rage Pro	(AGP/PCI)
ATI Xpert@Work	2MB	3D Rage Pro	(PCI)
ATI Pro Turbo+PC2TV	4MB	3D Rage II+DVD	(rev 154)
ATI 3D Xpression+	4MB	3D Rage II	(GT-B, SGRAM, rev 65)
ATI 3D Xpression+	2MB	3D Rage II	(GT-B, SDRAM, rev 65)
ATI 3D Xpression	2MB	3D Rage	(GT-A, rev 72)
ATI Video Xpression+	2MB	Mach64 VT-A3	(rev 8)
ATI Video Xpression	2MB	Mach64 VT-A4	(rev 72)
ATI Graphics Xpression	2MB	Mach64 CT	(rev 9)
ATI Graphics Xpression	2MB	Mach64 CT-C	(rev 65)
ATI Graphics Xpression	2MB	Mach64 CT-D	(rev 10)
ATI Graphics Xpression	2MB	Mach64 GX	(rev 1) with Chrontel8398 RAMDAC
ATI Graphics Pro Turbo	2MB	Mach64 GX	(rev 0) with 68860-B RAMDAC
ATI Graphics Pro Turbo	2MB	Mach64 CX	(rev 1) with AT&T20C408 RAMDAC
ATI WinTurbo	2MB	Mach64 GX	(rev 1) with 68860-C RAMDAC

6. Known Problems and Bug Reports

There are several known problems with the current version of the Mach64 X server. They include:

- Gamma correction is not currently supported. It will be supported in a future release.
- Screen blanking in 16bpp and 32bpp modes on certain Mach64 CT cards does not work.
- In doublescan modes, only the top half of the hardware cursor is displayed. The hardware cursor works fine in all other modes.
- With high refresh rates on certain cards (VT-A3 and CT-D) noise can become a problem in 32bpp mode. This usually only happens with refresh rates of 85Hz or greater and can be fixed by using a lower refresh rate (e.g., 72Hz or 75Hz).
- ISA cards with more than 12Mb of main memory cannot use the server due to the requirement of a video memory aperture. This a major project.

If you are experiencing problems, first check to make sure that you have the very latest available release (including beta releases). ATI releases new cards throughout the year. Each of these new cards require additional programming to support the new Mach64 chips, RAMDACs and clock chips that appear on them. The most recent release is most likely to support your video card.

Second, please check the RELNOTES and README files (as well as the other documentation available with the release). Third, make sure you do not have any Ramdac, ClockChip or Clocks lines in your XF86Config file (all of these are automatically detected by the Mach64 X server). The "Device" section should only contain the Identifier, VendorName and BoardName. All other options should be automatically detected.

If you are still experiencing problems, please send e-mail to XFree86@XFree86.org or post to the comp.windows.x.i386unix newsgroup.

Please do NOT send e-mail to me since the developers who answer e-mail sent to XFree86@XFree86.org are better able to answer most questions and I would like to spend my minimal free time working on new enhancements to the X server. Thanks!

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Mach64.sgml,v 3.15.2.7 1999/07/05
09:07:28 hohndel Exp $
```

```
$XConsortium: Mach64.sgml /main/8 1996/10/28 05:23:52 kaleb $
```

TOC

Information for NVidia NV1 / SGS-Thomson STG2000, Riva 128 and Riva TNT and TNT2 Users

David McKay, Dirk Hohndel

June 25 1999

1. Supported hardware

2. Notes

3. Authors

1. Supported hardware

This driver supports good acceleration for the NV1/STG2000 as well as the Riva128, Riva TNT and Riva TNT2.

2. Notes

- On the NV1/STG2000, the driver does not support the virtual desktop features of xfree86. This is because the NV1 does not have the necessary hardware to support this feature. If you want to change resolutions, you will have to modify your config file. Comment out all but the mode you wish to use.
- The generic VGA16 server will not work with the NV1. For this reason `XF86Setup` cannot be used to configure the server for NV1 based cards. Use `xf86config` instead. Select `Diamond Edge 3D' as your board, and select only **ONE** mode for each of 8bpp and 16bpp. Do not select a virtual desktop. Also, make sure you don't select a RAMDAC or clock chip. This does not apply if you own a Riva128 or RIVA TNT card, as the VGA16 server works just fine on that.
- Both the NV1 and the Riva128 only support a 555 RGB Weight in 16 bpp, the hardware does not do 565. If you run into problems with some window managers in 16bpp, try putting a Weight 555 in the Display section.
- 24 bpp is not supported.
- In some modes the hardware cursor gets out of sync with the display. Use Option "sw_cursor" to work around this problem.
- There are modelines that confuse the Riva128 chip. This results in a greenish display. Slightly modifying the modeline usually fixes the problem. In most cases all that is needed is to reduce the HTotal. You can use xvidtune to do that.
- The low maximum dot clocks for the Riva 128 have been fixed. The driver should now utilize the Riva 128 to its full capabilities.

3. Authors

- David McKay
- David Schmenk <*dschmenk@nvidia.com*>
- Dirk Hohndel <*hohndel@XFree86.org*>

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/NVIDIA.sgml,v 1.1.2.2 1999/06/25
08:57:14 hohndel Exp $
```

Information for Oak Technologies Inc. Chipset Users

Jorge F. Delgado Mendoza (*delgadomendoza.j@pg.com*)

17 August 1999

1. Supported chipsets

2. XF86Config options

3. Mode issues

4. Linear addressing

1. Supported chipsets

The driver is used in the 8-bit / 256-color SVGA server and the mono server. The following chipsets for Oak Tech. Inc. are supported:

OTI037C

8-bit VGA chipset, with up to 256Kbytes of DRAM. All the boards I have seen are only able to do standard VGA modes. (ie. up to 320x200x256 and up to 640x480x16). Currently the probe for this chip is disabled, so use the generic VGA driver instead.

OTI067

ISA SVGA chipset, up to 512Kbytes of DRAM (usually 70/80 ns).

OTI077

Enhanced version of the 067, with support for 1Mbyte and up to 65 Mhz dot-clock. This chipset is capable of resolutions up to 1024x768x256 colors in Non-Interlaced mode, and up to 1280x1024x16 colors Interlaced.

OTI087

One of the first VLB chipsets available, it has a 16-bit external data path, and a 32-bit internal memory-controller data path. It features some acceleration hardware: register-based color expansion, hardware cursor, a primitive BitBlt engine, a 64 bit graphic latch and some other new (on its time) features. Maximum BIOS resolutions are 1024x768x256 Non-Interlaced and 1280x1024x256 interlaced. Maximum Dot-Clock is 80Mhz, but is usually coupled with the OTI068 clock generator capable of frequencies up to 78Mhz. This chipset supports up to 2MBytes of 70/70R ns DRAM.

OTI107 and OTI111

These are new, PCI chipsets by Oak Tech. Inc. Support is not included for them, as they are very rare and I haven't had the chance to look at one of these boards. We have been unable to locate 107's. If anybody has such a board and can donate it to XFree86, we would be more than glad to add support for them.

An OTI111 is now available and we are working on support for it.

All the chipsets up to the OTI087 are "Backwards compatible", in fact some early drivers for the OTI087 based chipsets were those made for the 077.

Accelerated support is included only for OTI087 chipsets, also Mono server is only included for 067/077 chipsets.

Previous: *Information for Oak Technologies Inc. Chipset Users*
Next: *XF86Config options*

TOC

2. XF86Config options

The following options are of particular interest to the Oak driver. Each of them must be specified in the 'svga' driver section of the `XF86Config` file, within the Screen subsections to which they are applicable (you can enable options for all depths by specifying them in the Device section).

Option "linear" (OTI087)

This option enables a linear framebuffer at 0xE00000 (14Mb) for cards recognized as ISA by the probe. Cards that are VLB will map the framebuffer at 0x4E00000. The aperture depends on the VideoRam parameter in the `XF86Config` file or on the probed value for the board. It will speed up performance by about 15% on a VLB-based boards for a DX2-66 486.

Sometimes a motherboard will not be able to map at 0x4E00000, and then linear mode will not work with more than 14 Mbytes of main RAM. I know this because mine doesn't.

Option "fifo_aggressive" (OTI087)

This option will cause the command FIFO threshold of the chipset to be set at 0 instructions, which should be optimal for 16-bit data transfers, as empirical use of different thresholds, with xbench, show. Expect a 5-10% of performance boost on a DX2-66 486.

Option "fifo_conservative" (OTI087)

This option will set the FIFO to a safe value of 14, slowing the board by a 50%, use this only if you experience streaks or anomalies on the screen.

Option "enable_bitblt" (OTI087)

This option will enable an internal cache on the board that will be used as a rudimentary bitblt engine. Performance boost is more or less 100%, (double BlitStones on xbench). Most OTI087 boards seem to have this feature broken, corrupting text from xterms and leaving mouse droppings throughout the screen. As a rule of thumb, enable it, if it works badly, disable it.

Option "clock_50" (OTI087)

This one will force the internal speed to 50 Mhz.

Option "clock_66" (OTI087)

This one will force the internal speed to 66 Mhz, speeding up performance of the chipset.

Option "no_wait" (OTI087)

Sets the VLB interface to no wait states. On a medium VLB board (mine is VLB/PCI, so its not a very fast one) in VLB transparent mode, it manages up to 16 Mbytes/second transfer rate through the bus.

Option "first_wait" (OTI087)


TOC

Makes the VLB interface to add one wait state to the first read or write of a given burst.

Option "first_wwait" (OTI087)

Similar to the previous one, this only inserts a wait state in the first 'write' of a given burst. reads are not affected. This is the default behaviour of the server.

Option "write_wait" (OTI087)

This configures the VLB interface to add one wait state to each write cycle.

Option "read_wait" (OTI087)

This configures the VLB interface to add one wait state to each read cycle.

Option "all_wait" (OTI087)

Enables the slowest VLB transfer adding wait states in all cases. Hopefully, no board will need this enabled.

Option "one_wait" (OTI087)

Sets the VLB interface to at least one wait state.

Option "noaccel" (OTI087)

One accelerated routine has been lately added to the driver, allowing it to draw solid fills quite faster. This routine only works (up to date) on segmented addressing, and only if the virtual width is 1024. This option is automatically enabled by the driver. Use this option if you want to disable it.

As a rule of thumb, use the option "no_wait", and if it doesn't result in corrupting text, lucky you. If not, try "first_wwait", and downwards. ISA card owners should not use these options.

Information for Oak Technologies Inc. Chipset Users **:** *XF86Config options*
Previous: *Supported chipsets*
Next: *Mode issues*

3. Mode issues

The use of very high dot-clocks has a REAL negative effect on the performance of the boards, due to its limited 80Mbit/sec, higher dot clocks limit its ability to draw data into the framebuffer. Thus expect better performance of a 72Mhz based mode than on a 78Mhz based one (for example) where more bandwidth is required for screen refresh.

It does not make much sense to use the highest clock (78 MHz) for 1024x768 at 76 Hz on a OTI087; the card will almost come to a standstill. A 72 MHz dot clock results in 70 Hz which should be acceptable. If you have a monitor that supports 1024x768 at 76 Hz with a 78 MHz dot clock, a standard OTI087 based card is a poor match anyway.

TOC

4. Linear addressing

Linear addressing is hardwired to 14 Mbytes for ISA boards and 78 Mbytes for VLB boards, thus if you have more than that on your board you shouldn't enable it. The aperture is selected from the VideoRam parameter of the XF86Config or from the amount of memory that is detected if VideoRam is not found.

I hope (because I have not tested it very thoroughly) that linear addressing will work on all ISA boards, VLB ones work flawlessly.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Oak.sgml,v 3.12.2.4 1999/08/18
13:12:11 hohndel Exp $
```

```
$XConsortium: Oak.sgml /main/8 1996/05/12 20:58:00 kaleb $
```

XFree86 P9000 Server Release Notes

Erik Nygren (*nygren@mit.edu*)

1998 December 29

1. Change Log

2. Supported Cards

    2.1. Diamond Viper VLB

    2.2. Diamond Viper PCI

3. Orchid P9000 and random clones

4. Viper Pro and other P9100 and P9130 cards (UNSUPPORTED!!!)

5. Acceleration

6. XFree86-DGA Extension Support

7. High Color and TrueColor

8. Random Notes

9. Operating System Notes

    9.1. NetBSD

10. XF86Config

11. Known Bugs

12. Credits

1. Change Log

1998.10.13:

- Fixed a bug that would cause the server to crash when it tried to enable or disable the screen saver while at a VT (Erik Nygren)

1997.01.30:

- Added probing for MemBase and IOBase on Diamond Viper PCI cards (Karl Anders Øygard)
- Added support for DPMI screen saving (Karl Anders Øygard)

1996.03.31:

- Added support for the XFree86-DGA extension (Erik Nygren)

1995.05.24:

- Added p9000frect.c: Accelerated solid rectangle fills at 8/16bpp (Henrik Harmsen)
- Added stipple fills to p9000frect.c, and a stub for tile fills when p9000ImageFill is fixed (Chris Mason)
- Added p9000pntwin.c: Accelerated paint window at 8/16/32bpp (Henrik Harmsen)
- Added p9000gc16.c and p9000gc32.c for the higher bpp drawing functions (Henrik Harmsen)
- Additions to p9000im.c: p9000Image[Op]Stipple. And p9000ImageFill. Currently, there are small problems with ImageFill, and it is not being used. (Chris Mason)
- Added p9000PixAlu and p9000PixOpAlu. miniterm->alu translation for pixel1 opaque and transparent operations. (Chris Mason)
- Added p9000text.c: Non-cached poly text and image text functions. Image text functions are not used because they are too slow :((Chris Mason)

1995.05.21:

- Fixed p9000init.c to properly deal with the `vram_128` option. This should allow the driver to work properly with all Viper's with 1 MB of memory. (Erik Nygren)

1995.01.29:

- Updated P9000.sgml to mention using Robin's scanpci rather than PCIDUMP.EXE.

1995.01.15:

- Fixed problem with line capping in accelerated line drawing. (Chris Mason)
- Fixed p9000QuadAlu[GXset] to be ~0 rather than 1. (Erik Nygren)

1995.01.14:

- `Clocks` line is no longer used in `XF86Config` file. Operation should now be consistent

with the operation of the other servers which use programmable clocks. (Erik Nygren)

- Users with 1MB cards can now explicitly specify `videoRam` in the `XF86Config` file when autoprobing fails. The new `vram_128` option may also be used to force the detection of 128Kx8 SIMM's. (Erik Nygren)
- Added p9000line.c and p9000seg.c for accelerated line drawing code using the p9000 quad/clipping engine. Blazingly fast for 1 clipping rectangle, could be made faster for multiple clipping regions by using software clipping. There is still a bug which causes xtest to report `Cap style incorrect for thin line and CapNotLast` for the `XDrawLines` tests but not for the `XDrawLine` or `XDrawSegments` tests [fixed in 1995.01.15 patch]. (Chris Mason)
- Changed p9000blt.c, and p9000win.c to wait for the quad/blit engine to be free. Before a quad/blit, check SR_ISSUE_QBN, then blit, then when all blits are done, do a p9000QBNotBusy. (Chris Mason)
- Changed p9000init.c to clear the screen using the quad meta coord drawing mode. Appears the rect mode does not update the CINDEX register correctly. Changed the color to 1 (black) from 0. (Chris)
- Added p9000QuadAlu. When drawing a quad, the p9000 equivalent to X's source is the foreground mask. When bliting/pixel8ing/pixel1ing, it is the p9000 source mask and the p9000alu lookup table should be used. (Chris Mason)
- Added some more registers to p9000reg.h. (Chris Mason)

1994.09.20:

- Fixed problem which prevented 16 bpp modes from working (Erik Nygren)

1994.09.16:

- Added screen blanking support for 16 bpp and 32 bpp modes. Screen blanking now powers down the RAMDAC rather than just changing the planemask. (Chris Mason, Erik Nygren)
- Fixed more problems caused by switch to `XF86Config` (Erik Nygren)
- Possible fix to maxclock for Orchid P9000 (Harry Langenbacher, Erik Nygren)

1994.09.15:

- Now almost always works with `XF86Config` changes (Erik Nygren)
- Cursor code looks at VTSema before writing to RAMDAC. This had been causing the x11perf server crash (Erik Nygren)

1994.09.08:

- Fixed problem with xdm and restarting the server (Erik Nygren)
- Fixed and enabled ImageRead in CopyArea (Chris Mason)
- Made informational comments conform to standard :-) (Erik Nygren)

1994.09.05:

- Fixed BIOS probe for Viper PCI (Bob Hollinger)
- Fixes to Orchid P9000 support (Harry Langenbacher)
- Changing of datatypes in clock code (Harry Langenbacher)

- Fixed clock and misc reg restoration so now works fine with svgalib (Chris Mason, Har Erik)

1994.08.29:

- Increased number of memory regions in xf86_OSlib.h from 2 to 3 as needed by the Viper PCI (Erik Nygren)
- Changed method of short pauses in p9000vga.c to outb(0x80,0) (Erik)
- Rewrote routines to determine sysconfig from horizontal resolution. Also added check for valid hres to probe. (Erik Nygren)
- Added MoveWindow acceleration for all depths. Opaque move even looks nice at 32bpp now! (Chris Mason)
- Minor fixes to acceleration. Acceleration is now enabled by default (Chris Mason)
- Added `"noaccel"` option (Erik Nygren)
- Added some fixes for Viper PCI (Matt Thomas)

1994.07.21:

- Preliminary Viper PCI support - totally untested so disabled (Erik Nygren)
- Preliminary Orchid P9000 support - incomplete and totally untested so disabled (Erik Nygren)
- Preliminary accelerated support - incomplete and not fully tested so disabled (Erik Nygren and Chris Mason)

1994.07.08:

- 16 and 32 bpp TrueColor support (Erik Nygren)
- Color restoration hopefully fixed (Erik Nygren)
- Changes to how `"Modes"` line in Xconfig is processed
- Removed banking support :-(

2. Supported Cards

2.1. Diamond Viper VLB

All Viper VLB's should work with this server, hopefully... :-) Due to Diamond's putting the same BIOS in some Viper VLB's as are used in Viper PCI's, the probe may detect you have a Viper PCI when you really have a Viper VLB. If this happens, put `chipset "vipervlb"` into your `XF86Config` file.

2.2. Diamond Viper PCI

You may need to specify the chipset `"viperpci"` in your `XF86Config` file.

Previously you had to find out the values for `MemBase` and `IOBase` by yourself. These are now autodetected.

3. Orchid P9000 and random clones

The Orchid P9000 and other cards based on the Weitek board design (such as the STAR 2000) should now work. Talk to *harry@brain.jpl.nasa.gov* if you have problems with this. Specify the chipset `"orchid_p9000"` in the `Device` section of `XF86Config`

4. Viper Pro and other P9100 and P9130 cards (UNSUPPORTED!!!)

These are NOT supported yet by this server, but are supported in the p9x00 driver of the SVGA server.

XFree86 P9000 Server Release Notes : *Viper Pro and other P9100 and P9130 cards (UNSUPPORTED!!!)*
Previous: *Orchid P9000 and random clones*
Next: *Acceleration*

5. Acceleration

Some of the acceleration code is working, but there are probably still bugs. Only a very small number of accelerated features have been implemented. Before working on any acceleration, please contact *nygren@mit.edu* so we don't duplicate efforts. Acceleration may be turned off with the `"noaccel"` option. The following things are now accelerated:

- Hardware cursor (8/16/32bpp)
- MoveWindow (8/16/32bpp)
- CopyArea (8bpp)

6. XFree86-DGA Extension Support

The XFree86-DGA extension is now supported. Note that XF86DGASetViewPort command is not fully implemented due to hardware limitations of the P9000. The SetViewPort and SetVidPage commands have been hacked to allow double buffering under certain conditions.

For cards with 1MB or modes where xres*yres*Bpp > 1024K, no double buffering is supported. In this case, the bank size returned is equal to the amount of video memory. Using the XF86DGASetViewPort and XF86DGASetVidPage commands have no results.

For cards with 2MB and for modes where virtualX*virtualY*Bpp < 1024K, the behaviors of SetViewPort and SetVidPage are modified to allow double buffering. The bank size returned by XF86DGAGetVideo is equal to xres*yres*Bpp. In this mode, there are two buffers which can be written to, read from, and displayed. The XF86DGASetVidPage command can be used to switch between buffers 0 and 1 for I/O. Whichever buffer is selected will be available through the linear aperture with no offset. If XF86DGASetViewPort is called with ypos < yres, it will cause buffer 0 to be displayed. If ypos >= yres, buffer 1 will be displayed. The result of this behavior is that programs which switch banks as necessary and which use two vertically adjacent banks should work with no P9000-specific changes.

7. High Color and TrueColor

Support for 16 and 24 bit truecolor is now supported. Note that 24 bit color is really 32 bits per pixel. Use the `-bpp` option when starting the server. Examples:

```
startx -- -bpp 32
startx -- -bpp 16
startx -- -bpp 16 -weight 555
startx -- -bpp 16 -weight 565
```

Note that many programs do not yet work properly with these modes. Don't tell me. Tell the authors unless they've already fixed it. It's their fault... :-)

Example problems:

xv 3.00

Works fine in 32 bpp and in 16 bpp with 24 bit images. Has problems with colors in 8 bit images in 8 bpp mode.

Mosaic 2.1

Has problems with colormap in both 16 bpp and 32 bpp. Newer versions of Mosaic such as 2.4 do work.

mpeg_play

Doesn't work at all in 16 bpp mode. Works fine 24 bpp mode when compiled with -DRS6000 and when run with ``-dither color"

xpaint 2.1

Works great in both modes but has a bug in the color requester for the selection tool. I think later versions may have fixed this.

8. Random Notes

Text restoration should now be fixed. Color restoration should also be fixed. You can now even run the server at the same time as svgalib programs!!!

Diamond has actually been fairly open and helpful. No NDA's were signed by anyone who wrote code and Diamond claims that none of the information they provided is proprietary.

One unresolved issue is the maximum clock speed. It is currently set to 135 MHz with a warning printed over 110 MHz. Diamond claims that this is the max in their docs, but examination has shown some Viper's to contain 110 MHz bt485's. Without 135 MHz, it is not possible for people to with large monitors to run at 1280x1024. Diamond claims that all Vipers have 135MHz bt485's or compatibles. If you have something slower, call their tech support and they will send you a RMA to get the board replaced.

9. Operating System Notes

Any operating system that can memory map linear regions in really high memory should work. This should include Linux, FreeBSD, SVR4, and more.

9.1. NetBSD

If you have NetBSD, you will need to install the aperture driver. Extract the file `apNetBSD.shar` (in `xc/programs/Xserver/hw/xfree86/etc/apNetBSD.shar`) and read the `README` contained therein.

10. XF86Config

The `modes` line in the `XF86Config` file is now handled differently. The virtual line is now ignored entirely. Each mode on the mode line is looked at and the first usable mode is selected (ie the first one which works with available memory, etc). Any other modes which are valid and have the same dimensions are also used. And other modes are ignored.

The current supported keywords in the `Device` section of the `XF86Config` file are:

VideoRAM

1024 or 2048 (use 2048 for ``3MB'' Orchid P9000's)

ChipSet

`"vipervlb"` or `"viperpci"` or `"orchid_p9000"`

MemBase

Viper VLB:

0xA0000000 or 0x20000000 or 0x80000000 (0x80000000 is default if none spec'd)

Orchid P9000:

0xC0000000 or 0xD0000000 or 0xE0000000 (this MUST be set to correspond to the jumpers)

Viper PCI:

any value corresponding to the output of `PCIDUMP.EXE`

IOBase

Viper PCI:

any value corresponding to the output of `PCIDUMP.EXE`

Others:

unused

Clocks

any values between 25 and 135 corresponding to the clocks for the mode entries being used. This line may now be omitted and clocks will be matched automatically.

Option

"sw_cursor"

use software cursor

"vram_128"

use if you have 1024K VRAM in 128Kx8 SIMMS

"sync_on_green"

generate sync pulses on the green signal. Most (all?) P9000 based boards don't support this.

"noaccel"

do not do hardware acceleration if it's causing problems for you

Modes

almost any valid mode (there are constraints on the horiz res so not all values are possible)

The current supported keywords in the Display section of the `XF86Config` file are:

Depth

8:

use 8 bits per pixel for 256 colors (default)

15 or 16:

use 16 bits per pixel for up to 65K colors

24 or 32:

use 32 bits per pixel (sparse 24 bpp) for up to 16 million colors

Weight

555 or 565 if Depth is 15 or 16. Otherwise this is ignored. These are the Red, Green, and Blue bits per pixel (default=565)

Here's a portion of a sample `XF86Config` file for the Viper VLB:

```
Section "Device"
    Identifier "ViperVLB"
    VendorName "Diamond"
    BoardName "Viper VLB"
    Videoram 2048                       # This is mandatory
    Membase 0x80000000                  # This is mandatory on non-ViperVLB's
    IOBase  0xe000                      # Use this ONLY on ViperPCI's
EndSection

Section "Screen"
    Driver "accel"
    Device "ViperVLB"
    Monitor "NEC4FGe"
    Subsection "Display"
        Depth 8   # This line is optional
        Modes "1024x768" "800x600"
    EndSubsection
EndSection
```

11. Known Bugs

There are currently problems with the server when used in conjunction with xdm, olvwm, and VT switching under Linux.

If the cursor changes while you're in a VT, the cursor won't look right when you return from the VT until it is moved between windows (and changes color and shape).

Memory probing does not work. You will need to explicitly specify the amount of memory you have. If you have a 1 MB card, try put `VideoRAM 1024` into the `Device` section of your `XF86Config` file. If this doesn't work, try adding `Option "vram_128"` to the `Device` section.

12. Credits

Major contributors to P9000 code:

- Erik Nygren (*nygren@mit.edu*)
- Harry Langenbacher (*harry@brain.jpl.nasa.gov*)
- Chris Mason (*clmtch@osfmail.isc.rit.edu*)
- Henrik Harmsen (*harmsen@eritel.se*)

Thanks to Matt Thomas (*thomas@lkg.dec.com*) and Bob Hollinger (*bob@interaccess.com*) for helping to get the Viper PCI server working.

Special thanks to David Moews (*dmoews@xraysgi.ims.uconn.edu*) whose banking patch could unfortunately not be included.

Thanks to Andy, David, Dave, Jon, Michael, Bob, all the XFree86 core team people, and everyone else!

During the course of the next few months, people will be working on acceleration, etc. Please send any patches to me (*nygren@mit.edu*).

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/P9000.sgml,v 3.18.2.7 1998/12/29
07:54:30 hohndel Exp $
```

```
$XConsortium: P9000.sgml /main/9 1996/05/12 20:58:05 kaleb $
```

Information for S3 Chipset Users

The XFree86 Project Inc.

27 February 1998

1. Supported hardware

2. 16bpp and 32bpp

3. List of Supported Clock Chips

4. List of Supported RAMDAC Chips

5. Additional Notes

6. Reference clock value for IBM RGB 5xx RAMDACs

7. Hints for LCD configuration (S3 Aurora64V+)

8. How to avoid ``snowing'' display while performing graphics operations

9. New S3 SVGA driver

1. Supported hardware

The current S3 Server supports the following S3 chipsets: 911, 924, 801/805, 928, 732 (Trio32), 764, 765, 775, 785 (Trio64*), 864, 868, 964, 968 and M65 (Aurora64V+). The S3 server will also recognise the 866, but it has not been tested with this chipset. If you have any problems or success with these, please report it to us.

Nevertheless, this is not enough to support every board using one of these chipsets. The following list contains some data points on boards that are known to work. If your card is similar to one of the described ones, chances are good it might work for you, too.

S3 801/805, AT&T 20C490 (or similar) RAMDAC

- Orchid Fahrenheit 1280+ VLB
- Actix GE32

8 and 15/16 bpp

Note: Real AT&T20C490 RAMDACs should be automatically detected by the server. For others which are compatible, you need to provide a `` `Ramdac "att20c490"' `` entry in your `XF86Config`.

Real AT&T 20C490 or 20C491 RAMDACs work with the `"dac_8_bit"` option. Some clones (like the Winbond 82C490) do not.

The Orchid Fahrenheit 1280+ VLB may require `` `Option "nolinear"' ``.

S3 805 VLB, S3 GENDAC (RAMDAC + clock synthesizer)

- MIRO 10SD (available for VLB and PCI) It is not known whether all 10SDs use the S3 GENDAC.

8 and 15/16 bpp

```
ClockChip "s3gendac"
RamDac    "s3gendac"
```

S3 801/805, AT&T 20C490 RAMDAC, ICD2061A Clockchip

- STB PowerGraph X.24 S3 (ISA)

8 and 15/16 bpp

Note: Real AT&T20C490 RAMDACs should be automatically detected by the server. For others which are compatible, you need to provide a `` `Ramdac "att20c490"' `` entry in your `XF86Config`.


```
ClockChip "icd2061a"
RamDac    "att20c490"
Option    "dac_8_bit
```

S3 805, Diamond SS2410 RAMDAC, ICD2061A Clockchip

- Diamond Stealth 24 VLB

8 and 15bpp(*) only.

requires `Option "nolinear"'

(*) The SS2410 RAMDAC is reportedly compatible with the AT&T20C490 in 15bpp mode. To make the server treat it as an AT&T20C490, you need to provide a `Ramdac "att20c490"' entry in your XF86Config.

S3 801/805, Chrontel 8391 Clockchip/Ramdac

- JAX 8241
- SPEA Mirage

8 and 15/16 bpp.

The 8391 is compatible with the AT&T 20C490 RAMDAC

```
ClockChip "ch8391"
Ramdac    "ch8391"
Option    "dac_8_bit"
```

S3 928, AT&T 20C490 RAMDAC

- Actix Ultra

8 and 15/16 bpp

Note: Real AT&T20C490 RAMDACs should be automatically detected by the server. For others which are compatible, you need to provide a `Ramdac "att20c490"' entry in your XF86Config. Also, the server's RAMDAC probe reportedly causes problems with some of these boards, and a RamDac entry should be used to avoid the probe.

Real AT&T 20C490 or 20C491 RAMDACs work with the "dac_8_bit" option. Some clones (like the Winbond 82C490) do not.

S3 928, Sierra SC15025 RAMDAC, ICD2061A Clockchip

- ELSA Winner 1000 ISA/EISA (``TwinBus", not Winner1000ISA!!)
- ELSA Winner 1000 VL

8, 15/16 and 24(32) bpp

Supports 8bit/pixel RGB in 8bpp and gamma correction for 15/16 and 24bpp modes


TOC

24 bpp might get ``snowy'' if the clock is near the limit of 30MHz. This is not considered dangerous, but limits the usability of 24 bpp.

D-step (or below) chips cannot be used with a line width of 1152; hence the most effective mode for a 1 MB board is about 1088x800x8 (similar to 2 MB, 1088x800x16).

```
ClockChip "icd2061a"
```

S3 928, Bt9485 RAMDAC, ICD2061A Clockchip

- STB Pegasus VL

8, 15/16 and 24(32) bpp

Supports RGB with sync-on-green if `"sync_on_green"` option is provided and board jumper is set for BNC outputs.

VLB linear addressing now occurs at 0x7FCxxxxx so that 64MB or more main memory can be supported without losing linear frame buffer access.

```
ClockChip "icd2061a"
Option    "stb_pegasus"
```

S3 928, Bt485 RAMDAC, SC11412 Clockchip

- SPEA Mercury 2MB VL

8, 15/16 and 24(32) bpp

```
ClockChip "SC11412"
Option    "SPEA_Mercury"
```

S3 928, Bt485 RAMDAC, ICD2061A Clockchip

- #9 GXE Level 10, 11, 12

8, 15/16 and 24(32) bpp

```
ClockChip "icd2061a"
Option    "number_nine"
```

S3 928, Ti3020 RAMDAC, ICD2061A Clockchip

- #9 GXE Level 14, 16

8, 15/16 and 24(32) bpp

Supports RGB with sync-on-green

TOC

```
        ClockChip "icd2061a"
        Option    "number_nine"
```

S3 864, AT&T20C498, ICS2494 Clockchip

- MIRO 20SD (BIOS 1.xx)

The ICS2494 is a fixed frequency clockchip, you have to use X -probeonly (without a Clocks line in XF86Config) to get the correct clock values.

8, 15/16 and 24(32) bpp

S3 864, AT&T20C498 or STG1700 RAMDAC, ICD2061A or ICS9161 Clockchip

- Elsa Winner1000PRO VLB
- Elsa Winner1000PRO PCI
- MIRO 20SD (BIOS 2.xx)
- Actix GraphicsENGINE 64 VLB/2MB

8, 15/16 and 24(32) bpp

```
        ClockChip "icd2061a"
```

S3 864, 20C498 or 21C498 RAMDAC, ICS2595 Clockchip

- SPEA MirageP64 2MB DRAM (BIOS 3.xx)

8, 15/16 and 24(32) bpp

Clockchip support is still sometimes flaky and on some machines problems with the first mode after startup of XF86_S3 or after switching back from VT have been seen; switching to next mode with CTRL+ALT+``KP+" and back seems to solve this problem.

Interlaced modes don't work correctly.

Mirage P64 with BIOS 4.xx uses the S3 SDAC.

```
        ClockChip "ics2595"
```

S3 864, S3 86C716 SDAC RAMDAC and Clockchip

- Elsa Winner1000PRO
- MIRO 20SD (BIOS 3.xx)
- SPEA MirageP64 2MB DRAM (BIOS 4.xx)
- Diamond Stealth 64 DRAM

8, 15/16 and 24 bpp

S3 864, ICS5342 RAMDAC and Clockchip

- Diamond Stealth 64 DRAM (only some cards)

8, 15/16 and 24 bpp

```
ClockChip "ics5342"
Ramdac    "ics5342"
```

S3 864, AT&T21C498-13 RAMDAC, ICD2061A Clockchip

- #9 GXE64 - PCI

8, 15/16, 24(32) bpp

```
ClockChip "icd2061a"
Option    "number_nine"
```

S3 964, AT&T 20C505 RAMDAC, ICD2061A Clockchip

- Miro Crystal 20SV

8, 15/16, 24(32) bpp

```
ClockChip "icd2061a"
Ramdac    "att20c505"
```

S3 964, Bt485 RAMDAC, ICD2061A Clockchip

- Diamond Stealth 64

8, 15/16, 24(32) bpp

```
ClockChip "icd2061a"
```

S3 964, Bt9485 or AT&T 20C505 RAMDAC, ICS9161a Clockchip

- SPEA Mercury 64

8, 15/16, 24(32) bpp

```
ClockChip "ics9161a"
Option    "SPEA_Mercury"
```

S3 964, Ti3020 RAMDAC, ICD2061A Clockchip

- Elsa Winner2000PRO PCI

8, 15/16, 24(32) bpp

```
ClockChip "icd2061a"
```

TOC

S3 964, Ti3025 RAMDAC, Ti3025 Clockchip

- Miro Crystal 40SV
- #9 GXE64 Pro VLB
- #9 GXE64 Pro PCI

8 bpp, 15, 16 and 24(32) bpp

There are some known problems with the GXE64 Pro support, including some image shifting/wrapping at 15/16/24 bpp.

We have found that #9 no longer support the GXE64 Pro at 1600x1200. They do however have a new (and more expensive) board called the GXE64Pro-1600 which uses a 220MHz RAMDAC instead of 135MHz part used on the other boards.

S3 764 (Trio64)

- SPEA Mirage P64 (BIOS 5.xx)
- Diamond Stealth 64 DRAM
- #9 GXE64 Trio64

8/15/16/24 bpp

Note: The Trio64 has a builtin RAMDAC and clockchip, so the server should work with all Trio64 cards, and there is no need to specify the RAMDAC or clockchip in the `XF86Config` file.

S3 732 (Trio32)

- Diamond Stealth 64 DRAM SE

8/15/16/24 bpp

Note: The Trio32 has a builtin RAMDAC and clockchip, so the server should work with all Trio32 cards, and there is no need to specify the RAMDAC or clockchip in the `XF86Config` file.

S3 868, S3 86C716 SDAC RAMDAC and Clockchip

- ELSA Winner 1000AVI
- Diamond Stealth Video DRAM

8/15/16/24 bpp

S3 868, AT&T 20C409 RAMDAC and Clockchip

- ELSA Winner 1000AVI

8/15/16/24 bpp

Note: pixelmultiplexing is not supported yet, therefore limited maximum dot clock for 8bpp (currently 67.5MHz, should be changed to 100MHz if pixmux isn't fixed prior to release)

S3 968, Ti3026 RAMDAC, Ti3026 Clockchip

- Elsa Winner 2000PRO/X-2 and /X-4 (Revsions <= F)
- Elsa Winner 2000AVI-2 and -4
- Diamond Stealth 64 VIDEO VRAM


TOC

8/15/16/24 bpp

S3 968, Ti3026 RAMDAC, ICS9161A Clockchip

- Elsa Winner 2000PRO/X-2 and /X-4 (Revision G)

8/15/16/24 bpp

Note: clock doubling doesn't work, yet, therefore the maximum usable dot clock is limited to about 120MHz.

S3 964, IBM RGB 514/524/525/528 RAMDAC & Clockchip

- Hercules Graphics Terminator 64

8/15/16/24 bpp

```
s3RefClk   50
DACspeed  170
Option  "slow_vram"
```

S3 968, IBM RGB 514/524/525/528 RAMDAC & Clockchip

- Genoa Genoa VideoBlitz III AV

```
s3RefClk   50
DACspeed  170
```

- Hercules Graphics Terminator Pro 64

```
s3RefClk   16
DACspeed  220
```

This card may require the line:

```
Invert_VCLK "*" 0
```

in each Display subsection.

- STB Velocity 64

```
s3RefClk   24
DACspeed  220
```

- Number Nine FX Motion 771

```
s3RefClk   16
DACspeed  220
```

This card may require the line:

```
Invert_VCLK "*" 0
```

in each Display subsection.

- MIRO 80SV

```
s3RefClk   16
DACspeed  250
```

8/15/16/24 bpp

ELSA Winner 2000PRO/X-8 (S3 968, 8MB VRAM, 220MHz for 32bpp)

The server has only been tested for "revision C" of this card (guess the serial number should start with C, but not sure since mine says Ser.No. A-0000.000.000;) which have an IBM RGB528A note the A; can't be probed though)

depending on the mode line etc there may be some display distortions like:

1. many long horizontal lines/stripes
2. pixel jitter or short horizontal stripes like snow all over the screen
3. Like 2., but only when doing graphics ops (like opaque move of windows).
4. additional pixel at the left display edge and some missing pixels at the right edge.

All of these problems can be fixed by small adjustments to the mode line (best to run `xvidtune' and make these adjustments interactively). E.g., for the first three problems, shift the display left or right a few steps. For the last problem, increasing HSyncEnd (making the hsync pulse longer) solves the problem. In some cases, a significant increase in the sync pulse width is needed, and rarely, it needs to be shortened (by decreasing HSyncEnd).

In rare cases, InvertVCLK and/or EarlySC may need to be adjusted, followed by an adjustment of BlankDelay (see the bottom line of xvidtune).

If you see any of these problems, please contact koenig@XFree86.org, and send details of:

- Original mode showing the problem,
- Tuned/fixed mode, including all flags from the bottom line of xvidtune,
- Colour depth used for this tuned mode line,
- Full server startup output.

2. 16bpp and 32bpp

On 801/805 + AT&T490 Cards (like the Fahrenheit 1280+ VLB) only 15 and 16bpp are supported. 32bpp isn't available on this type of card. (There is a 24 bit mode under MS Windows, but it's not a 32bpp sparse mode but a real 3 bytes/pixel mode).

3. List of Supported Clock Chips

```
ICD2061A                                ==> ClockChip "icd2061a"
ICS9161A (ICD2061A compatible)          ==> ClockChip "ics9161a"
DCS2824-0 (Diamond, ICD2061A comp.)     ==> ClockChip "dcs2824"

S3 86c708 GENDAC                        ==> ClockChip "s3gendac"
ICS5300 GENDAC (86c708 compatible)      ==> ClockChip "ics5300"

S3 86c716 SDAC                          ==> ClockChip "s3_sdac"
ICS5342 GENDAC                          ==> ClockChip "ics5342"
STG 1703                                ==> ClockChip "stg1703"

Sierra SC11412                          ==> ClockChip "sc11412"
ICS2595                                 ==> ClockChip "ics2595"
TI3025                                  ==> ClockChip "ti3025"
TI3026                                  ==> ClockChip "ti3026"
IBM RGB 5xx                             ==> ClockChip "ibm_rgb5xx"

Chrontel 8391                           ==> ClockChip "ch8391"

AT&T 20C409                             ==> ClockChip "att20c409"
AT&T 20C499 (untested)          ==> ClockChip "att20c499"
```

4. List of Supported RAMDAC Chips

If you have a RAMDAC that is not listed here, be VERY careful not to overdrive it using XF86_S3. Better contact the XFree86 team first to verify that running XF86_S3 will not damage your board.

RAMDACs that are grouped together below are treated as compatible with each other as far as the server is concerned. For example, the server will report `"bt485"` when you actually specify `RAMDAC "bt9485"`, or `"s3_gendac"` when you specify `RAMDAC "ics5300"`.

```
ATT20C409                               ==> RAMDAC "att20c409"

ATT20C490                               ==> RAMDAC "att20c490"
ATT20C491                               ==> RAMDAC "att20c491"
CH8391                                  ==> RAMDAC "ch8391"

ATT20C498                               ==> RAMDAC "att20c498"
ATT21C498                               ==> RAMDAC "att21c498"

ATT22C498                               ==> RAMDAC "att22c498"

ATT20C505                               ==> RAMDAC "att20c505"

BT485                                   ==> RAMDAC "bt485"
BT9485                                  ==> RAMDAC "bt9485"

IBMRGB514                               ==> RAMDAC "ibm_rgb514"
IBMRGB525                               ==> RAMDAC "ibm_rgb525"

IBMRGB524                               ==> RAMDAC "ibm_rgb524"
IBMRGB526                               ==> RAMDAC "ibm_rgb526"

IBMRGB528                               ==> RAMDAC "ibm_rgb528"

S3_GENDAC                               ==> RAMDAC "s3gendac"
ICS5300                                 ==> RAMDAC "ics5300"

S3_SDAC                                 ==> RAMDAC "s3_sdac"
ICS5342                                 ==> RAMDAC "ics5342"

S3_TRIO32                               ==> RAMDAC "s3_trio32"
```


```
S3_TRIO64                                   ==> RAMDAC "s3_trio64"
S3_TRIO64                                   ==> RAMDAC "s3_trio"

SC11482                                     ==> RAMDAC "sc11482"
SC11483                                     ==> RAMDAC "sc11483"
SC11484                                     ==> RAMDAC "sc11484"

SC11485                                     ==> RAMDAC "sc11485"
SC11487                                     ==> RAMDAC "sc11487"
SC11489                                     ==> RAMDAC "sc11489"

SC15025                                     ==> RAMDAC "sc15025"

STG1700                                     ==> RAMDAC "stg1700"

STG1703                                     ==> RAMDAC "stg1703"

TI3020                                      ==> RAMDAC "ti3020"

TI3025                                      ==> RAMDAC "ti3025"

TI3026                                      ==> RAMDAC "ti3026"

None of the above                           ==> RAMDAC "normal"
```

If you feel adventurous you could also open up your computer and have a peek at your RAMDAC. The RAMDAC is usually one of the larger chips (second or third largest chip that is NOT an EPROM) on the board. The markings on it are usually

```
  <Company logo>

      <company identifier><part number>-<speed grade>
      <manufacturing week><manufacturing year>
      <lot number><other funny numbers>
```

For example:

```
        @@
        @@ AT&T

      ATT20C490-11
       9339S ES
        9869874
```

This is an AT&T 20C490 with a speed grade of 110 MHz. This would then mean that you put a

`DacSpeed 110' line in your `XF86Config` file. Be advised that some RAMDACs have different modes that have different limits. The manufacturer will always mark the chip naming the higher limits, so you should be careful. The S3 server knows how to handle the limits for most of the RAMDACs it supports providing the DacSpeed is specified correctly.

Chips labeled **-80** or **-8** should use `DacSpeed 80' in the device section.

```
S3 86C716-ME SDAC  ==>  DacSpeed 110
SC15025-8          ==>  DacSpeed  80
ATT20C490-80       ==>  DacSpeed  80

IBM 8190429        ==>  DacSpeed 170
IBM 03H5428        ==>  DacSpeed 170
IBM 03H6447        ==>  DacSpeed 170
IBM 03H6448        ==>  DacSpeed 220
IBM 03H5319        ==>  DacSpeed 220
IBM 63G9902        ==>  DacSpeed 250

IBM 37RGB514CF17   ==>  DacSpeed 170
IBM 37RGB524CF22   ==>  DacSpeed 220
                ^
```

Information for S3 Chipset Users **:** *List of Supported RAMDAC Chips*
Previous: *List of Supported Clock Chips*
Next: *Additional Notes*

5. Additional Notes

Note that the Sierra SC1148{5,7,9} will not be distinguished from the Sierra SC1148{2,3,4} by the probe. The only difference between the two series as far as the server is concerned is that the {2,3,4} is capable of 15bpp, while the {5,7,9} is capable of 16bpp. So if you have a SC1148{5,7,9} and want to use 16bpp instead of 15bpp, you will have to specify a `RAMDAC "sc11485"` line as shown above.

Some RAMDACs (like the Ti3025) require some mode timing consideration for their hardware cursor to work correctly. The Ti3025 requires that the mode have a back porch of at least 80 pixel-clock cycles. A symptom of this not being correct is the HW cursor being chopped off when positioned close to the right edge of the screen.

6. Reference clock value for IBM RGB 5xx RAMDACs

Cards with IBM RGB5xx RAMDACs use several different input frequencies for the clock synthesizer which can't be probed without some knowledge of the text mode clocks (which may be a wrong assumption if you're using non-standard text modes). Here is the procedure you should use to find out the input frequency:

First run

```
X -probeonly >& outfile
```

and check the output for the probed clock chip which might look like this:

```
(--) S3: Using IBM RGB52x programmable clock (MCLK 66.000 MHz)
(--) S3: with refclock 16.000 MHz (probed 15.952 & 16.041)
                       ^^^^^^      ^^^^^^^^^^^^^^^^^^^^
```

there will be a "good guessed" value which will be used and two probed values in brackets based on the 25MHz and 28MHz text clocks. This probing can only work if you run a normal 80x25 or 80x28 text mode!

The refclock values known so far are:

```
STB Velocity 64          24 Mhz
Genoa VideoBlitz II AV   50 MHz
Hercules S3 964          50 MHz
Hercules S3 968          16 MHz
#9 Motion 771            16 MHz
```

depending on the quartz on your card and maybe other features like an additional clock chip on the Genoa card (which as a 14.3MHz quartz).

If you claim that your card has a 16MHz clock, but it really uses 50MHz, all pixel clocks will be tripled and a 640x480 mode with 25MHz will use a 75MHz pixel clock, so be very careful.

If you found the right refclock, you should set it in the config file (device section) e.g. with

```
s3RefClk  16
```

or


```
s3RefClk  50
```

so that that this value will be used even if you use another text mode and probing fails!

Information for S3 Chipset Users **:** *Reference clock value for IBM RGB 5xx RAMDACs*
Previous: *Additional Notes*
Next: *Hints for LCD configuration (S3 Aurora64V+)*

7. Hints for LCD configuration (S3 Aurora64V+)

If LCD is active the CRT will always output 1024x768 (or whatever is the _physical_ LCD size) and smaller modes are zoomed to fit on the LCD unless you specify Option "lcd_center" in the device section.

The pixel clock for this physical size (e.g. 1024x768) mode...

- ...can explicitly set in the config file (device section) with e.g. `Set_LCDClk 70' (resulting 70 MHz pixel clock being used for all modes when LCD is on)
- ...is taken from the _first_ mode in the modes line iff this mode's display size is the same as the physical LCD size
- ...the default LCD pixel clock of BIOS initialisation setup is used. This value is output at server startup in the line `LCD size ...' unless you're specifying a value using `Set_LCDClk ...'

If LCD is _not_ active, the normal mode lines and pixel clocks are used for the VGA output.

Whenever you switch output sources with Fn-F5, the Xserver won't get informed and pixel clock and other settings are wrong. Because of this you have to switch modes _after_ switch output sources! Then the server will check which outputs are active and select the correct clocks etc. So the recommended key sequence to switch output is

Fn-F5 Ctrl-Alt-Plus Ctrl-Alt-Minus

and everything should be ok..

on the Toshiba keypad you can first hold down Ctrl-Alt, then press `Fn' additionally before pressing Plus/Minus too to avoid to explicitly enable/disable the numeric keypad for mode switching.

8. How to avoid ``snowing'' display while performing graphics operations

For cards with the S3 Vision864 chip, there is an automatic correction which depends on the pixel clock and the memory clock MCLK at which the S3 chip operates. For most clock chips this value can't be read (only the S3 SDAC allows reading the MCLK value so far), so this value has to be estimated and specified by the user (the default is 60 [MHz]).

With the new `s3MCLK' entry for your `XF86Config` file, now you can specify e.g.

```
s3MCLK 55
```

for a 55 MHz MCLK which will reduce snowing. Smaller MCLK values will reduce performance a bit so you shouldn't use a too low value (55 or 50 should be a good guess in most cases).

Below is a small shell script which might be useful to determine the approximate value for MCLK (about +/- 1-2 MHz error). Before running this script you have to add the line

```
s3MNadjust -31 255
```

to the device section in your `XF86Config` file and restart X Windows. With this option (which is for testing and debugging only) you'll get lots of disastrous display flickering and snowing, so it should be removed again immediately after running the test script below.

Running this script will use xbench and/or x11perf to run a test to determine the MLCK value, which is printed in MHz. Up to 4 tests are run, so you'll get up to 4 estimates (where the first might be the most accurate one).

```
#!/bin/sh

exec 2> /dev/null

scale=2

calc() {
  m=`awk 'BEGIN{printf "%.'$scale'f\n",'"( $1 + $2 ) / $3; exit}" `
  [ -z "$m" ] && m=` echo "scale=$scale; ( $1 + $2 ) / $3" | bc `
  [ -z "$m" ] && m=` echo "$scale $1 $2 + $3 / pq" | dc `
  echo $m
}

run_xbench() {
  r=` ( echo 1; echo 2; echo 3; echo 4 ) | xbench -only $1 | grep rate `
```


```
  [ -z "$r" ] && return
  cp="$2 $3"
  set $r
  calc $3 $cp
}

run_x11perf() {
  r=` x11perf $1 | grep trep | tr '(/)' '   ' `
  [ -z "$r" ] && return
  cp="$2 $3"
  set $r
  calc `calc 1000 0 $4` $cp
}

run_x11perf "-rect500 -rop GXxor"     3.86  5.53  #  0 1  #    4.11    5.52  #
run_xbench invrects500                4.63  5.48  #  0 1  #    4.69    5.48  #

run_x11perf "-rect500 -rop GXcopy"  -16.42 13.90  #  0 1  #  -14.99   13.88  #
run_xbench fillrects500              -7.81 13.57  #  0 1  #   -8.53   13.58  #

exit
```

Information for S3 Chipset Users : *How to avoid ``snowing'' display while performing graphics operations*
Previous: *Hints for LCD configuration (S3 Aurora64V+)*
Next: *New S3 SVGA driver*

9. New S3 SVGA driver

There is a new experimental S3 driver for non-ViRGE S3 chipsets in the XF86_SVGA server. This is definitely an ALPHA quality driver and hasn't been well tested, and has some known problems. Because of this, the configuration programs will install XF86_S3 by default rather than this one. But if you're adventurous or had some problems with XF86_S3, you might want to give it a try.

The driver includes generic S3 support which should work on all non-ViRGE S3 chips (in theory, that is). It also has improved support for chips that support S3's new style memory mapped I/O. These chips include the 868, 968 and recent Trio64 variants (not the plain old Trio64s). Chips that are capable of using the new style MMIO will use it automatically. The option "NO_MMIO" can be used to turn this off.

Performance for chips using the new style MMIO is expected to be better than XF86_S3, especially on a PCI bus. Performance without MMIO, however, is expected to be roughly comparable to XF86_S3 (faster in some areas, slower in others).

All color depths achievable with XF86_S3 should be possible with these drivers. Additionally, packed 24 bpp "sort of" works for the 868 and 968. Your results may vary.

Nearly all the options and features supported by XF86_S3 are supported by this driver. Additionally, the standard XAA/SVGA server options such as NO_ACCEL, SW_CURSOR, and NO_PIXMAP_CACHE are also supported. XF86_S3 features which are NOT supported in this driver are DPMS support and gamma correction.

The driver supports the PCI_RETRY option when using MMIO and a PCI card. This option can give large performance boosts for some operations, but has a tendency to hog the bus. Because of this, the option is not set by default. Most hardware combinations may not have any problems using this option, but sound card glitches during intensive graphics operations have been reported on some.

One shortcoming worth noting is that this driver does not yet contain the work-around for some S3 PCI BIOSs that report their memory usage incorrectly. This can result in conflicting address spaces. If this is the case on your hardware you should run XF86_S3 once and write down the address that your card is relocated to (as printed out in the server output). Then you can force the server to use this address with the MemBase field in the XF86Config (see the man page on XF86Config).

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/S3.sgml,v 3.37.2.4 1998/02/27
02:34:38 dawes Exp $
```

```
$XConsortium: S3.sgml /main/14 1996/02/21 17:45:58 kaleb $
```

Information for S3 ViRGE, ViRGE/DX, ViRGE/GX, ViRGE/GX2, ViRGE/MX, ViRGE/VX, Trio3D, Trio3D/2X, Savage3D and Savage4 Users

The XFree86 Project Inc.

2 August 1999

1. Supported hardware

2. XF86_S3V server

   2.1. Features:

   2.2. Known limitations

   2.3. Configuration:

3. XF86_SVGA server

   3.1. Features

   3.2. Known limitations in the Savage family support (s3_savage driver)

   3.3. Known limitations of the s3_virge driver

   3.4. Configuration

   3.5. Hints for LCD configuration (S3 ViRGE/MX)

4. Authors

   4.1. XF86_S3V server

   4.2. XF86_SVGA ViRGE driver

1. Supported hardware

Since release 3.3.2 of XFree86, there are now two servers which support the ViRGE family of chips. The XF86_S3V server is a dedicated server which supports the S3 ViRGE (86C325), the ViRGE/DX (86C375), ViRGE/GX (86C385) and the ViRGE/VX (86C988) chips. Use of that server is no longer recommended. It is not actively being supported anymore.

The above ViRGE chipsets are supported in the XF86_SVGA server, which includes a new ViRGE driver making use of the XAA acceleration architecture and also supports ViRGE/GX2 (86C357), ViRGE/MX (86C260), Trio3D (86C365), Trio3D/2X (86C362), Savage3D (86C391) and Savage4 (86C396/86C397) chips as of 3.3.5.

The following sections describe details of ViRGE support. Be aware that there are two servers described. XF86_S3V is the ViRGE specific server and was created first. The new acceleration architecture support is found in the XF86_SVGA server using the s3_virge driver. Each has strengths and weaknesses.

2. XF86_S3V server

The S3V server has some minor fixes since 3.3.1. You should find that the ViRGE server is stable at all depths. The server supports 1 and 32 bpp pixmap formats. This fixes known problems with xanim and Netscape clients in early versions of the S3V server. It has been tested with ViRGE cards with 2 and 4MB DRAM, ViRGE/DX 4M, ViRGE/VX 8M (4M VRAM/4M DRAM), and with a 220MHz ViRGE/VX card with 2MB VRAM up to 1600x1200 with 8/15/16bpp.

NOTE: This driver is pretty new, and not everything might work like you expect it to. It shouldn't crash your machine, but you may have video artifacts or missing lines. Please report any and all problems to XFree86@Xfree86.org using the appropriate bug report sheet.

2.1. Features:

- Basic support for S3 ViRGE, ViRGE/DX, ViRGE/GX and ViRGE/VX video adapters
- uses linear frame buffer
- it should be possible to reach resolutions up to the maximum supported by your video card memory. (eg. 1600x1200 at 8 and 16bpp, 1280x1024 at 24/32 bpp for a 4 Meg. card)
- it should be possible to use pixel depths of 8, 15, 16, 24, and 32 bits per pixel.
- 32 bpp is implemented as translation to 24 bpp

2.2. Known limitations

- No support for external RAMDACs on the ViRGE/VX.
- No support for VLB cards.
- No support for doublescan modes.
- The driver only works with linear addressing.
- For 24/32 bpp some simple dashed line acceleration is implemented, but sloped dash/double dash are drawn as solid lines.
- No support for current chipsets.
- No longer actively maintained.

2.3. Configuration:

The server auto-detects RAM size, RAMDAC and ClockChip. Do not bother putting these in your "Device" section. The "nolinear" option is unsupported.

TOC

2.3.1. Cursor:

- The default is hardware cursor, no option is needed.
- "sw_cursor" switches to software cursor.

---

3. XF86_SVGA server

The XF86_SVGA ViRGE driver supports all current flavors of the S3 ViRGE chipset including Trio3D and the Savage family. It uses the XAA acceleration architecture for acceleration, and allows color depths of 8, 15, 16, 24 and 32 bpp. It has been tested on several 2MB and 4MB ViRGE cards, a 4MB ViRGE/DX card, a ViRGE/VX card and a 4MB Trio3D card. Resolutions of up to 1600x1200 have been achieved. This is an early release of this driver, and not everything may work as expected. Please note that Trio3D support is an initial release and not very well tested. Please report any problems to XFree86@Xfree86.org using the appropriate bug report sheet.

3.1. Features

- Supports PCI hardware, ViRGE, ViRGE/DX, ViRGE/GX, ViRGE/GX2, ViRGE/MX, ViRGE/VX, Trio3D and the Savage family.
- Supports 8bpp, 15/16bpp, 24bpp and 32bpp.
- VT switching seems to work well, no corruption reported at all color depths.
- Acceleration is pretty complete: Screen-to-screen copy, solid rectangle fills, CPU-to-screen color expansion, 8x8 pattern mono and color fills. Currently, the color expansion appears to be substantially faster than the accel server due to the optimized XAA routines.
- Acceleration at 32bpp is limited: only ScreenToScreen bitblit and solid rectangles are supported. The ViRGE itself has no support for 32bpp acceleration, so the graphics engine is used in 16bpp mode.
- All modes include support for a hardware cursor.

3.2. Known limitations in the Savage family support (s3_savage driver)

The Savage family driver for the Savage3D and the Savage4 was donated to XFree86 by S3 very closely before the release of XFree86-3.3.5. The driver violates a few design principles and goals, but since there is massive demand for it, we decided to include it in XFree86-3.3.5.

The following issues and problems will be addressed for a future release of this driver:

- Only supported on Linux/x86 due to use of the Linux Real Mode Interface.
- Restricted to modes that the BIOS supports. The Modeline specified in the XF86Config file is being ignored.
- Limited acceleration.
- Limited testing.

3.3. Known limitations of the s3_virge driver

- No support for external RAMDACs on the ViRGE/VX.
- No support for VLB cards.
- No support for doublescan modes.
- The driver only works with linear addressing.
- Lines and polygons are not accelerated yet (but XAA still provides some acceleration in this respect).
- Burst Command Interface (BCI) support and 32bpp support not implemented for the Trio3D.
- Trio3D support only works for some modelines. Many of the standard modelines do not work (often slightly

modifying the dot clock works, though). The following two modelines seem to work reliably at 8bpp and 24bpp:

```
Modeline "1024x768"  75  1024 1048 1184 1328  768  771  777  806 -hsync -vsync
Modeline "1280x1024" 135 1280 1312 1416 1664  1024 1027 1030 1064
```

The following two modelines seem to work reliably at 16bpp:

```
Modeline "640x480"  45.80 640 672 768 864 480 488 494 530 -hsync -vsync
Modeline "800x600"  36      800  824  896 1024    600  601  603  625
```

3.4. Configuration

The ViRGE SVGA driver supports a large number of XF86Config options, which can be used to tune PCI behavior and improve performance.

Memory options:

- "slow_edodram" will switch the ViRGE to 2-cycle edo mode. Try this if you encounter pixel corruption on the ViRGE. Using this option will cause a large decrease in performance.
- "early_ras_precharge" and "late_ras_precharge" will modify the memory timings, and may fix pixel corruption on some cards. The default behavior is set by the BIOS, and is normally "late_ras_precharge".
- "set_mclk value" sets the video memory clock rate to 'value' (in MHz). The performance of the card is directly proportional to the memory clocking, so this may provide a performance increase. The BIOS setting for your card is printed at server start-up. Often, "low-cost" cards use the S3 default of 50MHz. This can often be exceeded with faster memory, some cards may function reliably at 60 or 65 MHz (even higher on some recent /DX and /GX cards). Note that S3 only officially supports an MCLK of 50MHz and XFree86 does not encourage exceeding those specs. *** Note: This option should not be preceded by the "Option" keyword!

Acceleration and graphic engine:

- "noaccel" turns off all acceleration
- "fifo_aggressive", "fifo_moderate" and "fifo_conservative" alter the settings for the threshold at which the pixel FIFO takes over the internal memory bus to refill itself. The smaller this threshold, the better the acceleration performance of the card. You may try the fastest setting ("aggressive") and move down if you encounter pixel corruption. The optimal setting will probably depend on dot-clock and on color depth. Note that specifying any of these options will also alter other memory settings which should increase performance, so you should at least use "fifo_conservative" (this uses the chip defaults).

PCI options:

- "pci_burst_on" will enable PCI burst mode. This should work on all but a few "broken" PCI chipsets, and will increase performance.
- "pci_retry" will allow the driver to rely on PCI Retry to program the ViRGE registers. "pci_burst_on" must be enabled for this to work. This will increase performance, especially for small fills/blits, because the driver does not have to poll the ViRGE before sending it commands to make sure it is ready. It should work on most recent PCI chipsets. A possible side-effect is that it may interfere with DMA operations on the PCI bus (e.g. sound cards, floppy drive).

Cursor:

- "hw_cursor" turns on the hardware cursor.

Color depth options and limitations:

- Pixel multiplexing is used above 80MHz for 8bpp on the ViRGE.
- 15bpp is supported, use "-bpp 15" as an option to the server.
- 24bpp is supported using the STREAMS engine.
- 32bpp uses STREAMS as well; however, because the ViRGE does not really support 32 bpp "natively", acceleration

is quite limited.

- Both 24bpp and 32bpp do not support interlace modes.
- 32bpp is limited to a width of < 1024 pixels. (1024x768 is not possible, even if you have the memory.) This is a hardware limit of ViRGE chips.

3.5. Hints for LCD configuration (S3 ViRGE/MX)

If LCD is active the CRT will always output 1024x768 (or whatever is the _physical_ LCD size) and smaller modes are zoomed to fit on the LCD unless you specify Option "lcd_center" in the device section.

The pixel clock for this physical size (e.g. 1024x768) mode...

- ...can explicitly set in the config file (device section) with e.g. `Set_LCDClk 70' (resulting 70 MHz pixel clock being used for all modes when LCD is on)
- ...is taken from the _first_ mode in the modes line iff this mode's display size is the same as the physical LCD size
- ...the default LCD pixel clock of BIOS initialisation setup is used. This value is output at server startup in the line `LCD size ...' unless you're specifying a value using `Set_LCDClk ...'

If LCD is _not_ active, the normal mode lines and pixel clocks are used for the VGA output.

Whenever you switch output sources with Fn-F5 or similar, the Xserver won't get informed and pixel clock and other settings are wrong. Because of this you have to switch modes _after_ switch output sources! Then the server will check which outputs are active and select the correct clocks etc. So the recommended key sequence to switch output is

Fn-F5 Ctrl-Alt-Plus Ctrl-Alt-Minus

and everything should be ok..

on the Toshiba keypad you can first hold down Ctrl-Alt, then press `Fn' additionally before pressing Plus/Minus too to avoid to explicitly enable/disable the numeric keypad for mode switching.

Information for S3 ViRGE, ViRGE/DX, ViRGE/GX, ViRGE/GX2, ViRGE/MX, ViRGE/VX, Trio3D, Trio3D/2X, Savage3D and Savage4 Users **:** *XF86_SVGA server*
Previous: *XF86_S3V server*
Next: *Authors*

4. Authors

4.1. XF86_S3V server

Harald Koenig <*koenig@tat.physik.uni-tuebingen.de*>

and:

- Kevin Brosius *Cobra@compuserve.com*
- Berry Dijk *berry_dijk@tasking.nl*
- Dirk Hohndel *hohndel@XFree86.Org*
- Huver Hu *huver@amgraf.com*
- Dirk Vangestel *gesteld@sh.bel.alcatel.be*

4.2. XF86_SVGA ViRGE driver

Sebastien Marineau <*marineau@genie.uottawa.ca*>

and:

- Harald Koenig <*koenig@tat.physik.uni-tuebingen.de*>
- Kevin Brosius *Cobra@compuserve.com*
- Sebastian Kloska *kloska@mpimp-golm.mpg.de*
- Alok Ladsariya *alok@XFree86.Org*
- Dirk Hohndel *hohndel@XFree86.Org*

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/S3V.sgml,v 3.3.2.12 1999/08/02
12:45:00 hohndel Exp $
```

Information for SiS Users

Xavier Ducoin (*xavier@rd.lectra.fr*)

June 25 1999

1. Introduction

2. Supported chips

3. XF86Config Options

4. Modelines

5. Troubleshooting

1. Introduction

This driver was primarily written for the SiS86c201. It also works on the 202 , 205 and 5597/5598 chips. Support for 6326, 530 and 620 has been added since. Some support for SiS86c215 and 225 was added as well. This support consists simply in identify it as 205, so probably 86c215 won't work with acceleration (is a cheap 205 without some features).

The driver supports many advanced features. These include:

- Linear Addressing
- 8/15/16/24 bits per pixel
- Fully programmable clocks are supported
- H/W cursor support
- BitBLT acceleration of many operations
- XAA support (XFree86 Acceleration Architecture)

2. Supported chips

SiS 86c201

(External hardware clock)

SiS 86c202, SiS 86c2x5, SiS 5597/5598, SiS 6326, SiS 530, SiS 620

(Internal clock synthesizer)

Color expansion is not supported by the engine in 16M-color graphic mode.

3. XF86Config Options

The following options are of particular interest for the SiS driver. Each of them must be specified in the `svga' driver section of the XF86Config file, within the Screen subsections of the depths to which they are applicable (you can enable options for all depths by specifying them in the Device section).

Option "set_mclk"

This option lets you to modify the memory clocking of your card. (only for 5597 and 6326) Modifying the memory timings can destroy the device, but usually the only ill effects of overclocking is to have some noise an drawing errors, but BE CAREFUL. Usually a little increment can improve the drawing speed, and allows also higher dotclocks. The server reports default memclock on starting messages, so take it as a base. Units are in MHZ.

Option "dac_speed"

This option lets you to modify the maximum allowed dotclock (only for 5597 and 6326). Without it, the server makes a conservative guess based on memory clock, speed and number of banks. If your monitor supports higher dotclocks and you know that your card can do it, give a try. If the speed is too high for your configuration (but not for your monitor), the effects can vary from some noise on screen to a black screen. Don't use speeds greater than 135 Mhz, (175 for 6326), even if your monitor supports the dot-clock.

Option "noaccel"

By default the XAA (XFree86 Acceleration Architecture) is used. This option will disable the use of the XAA and will enable the old BitBlt acceleration operations. (see below).

Option "hw_clocks"

On chips 86c202 and later, the default is to use the programmable clock for all clocks. It is possible to use the fixed clocks supported by the chip instead of using this option (manufacturer dependent).

Option "sw_cursor", "hw_cursor"

The default is for using the hardware cursor.

Option "no_linear"

By default linear addressing is used on all chips. However this might be broken in some implementations. It is possible to turn the linear addressing off with this option. Note that H/W acceleration and 16/24bpp are only supported with linear addressing.

Option "no_bitblt"

This option will disable the use of all the BitBLT engine. It is useful for problems related to acceleration problems. In general this will result in a reduced performance.

Option "no_imageblt"

It is useful for problems related to image writing, and possible stipple acceleration problems. In general this will result in a reduced performance.

Option "ext_eng_queue"

5597/8 and 6326 have the option to extend the engine command queue on VRAM. With extended queue length, the driver only checks queue status on some color-expansion commands. This gives some performance improvement, but is possible to lose some commands, corrupting screen output. As the size of extended command queue is 16-32K, the probability is very low, but exists. The performance gain observed is around 8-10%. Currently, using this option with xaa_benchmark freezes the acceleration engine, causing weird image display.

Option "pci_burst_on"

This set a bit on some registers. Although documented, the utility of this option is unknown for me. I can't see any difference on stability or performance.

Option "fast_vram"

Enables 1 cycle memory access. Try it. Increased memory bandwidth reduces the possibility of glitches and noise on high resolution modes.

Option "fifo_moderate","fifo_conservative","fifo_aggressive"

These options modify the arbitration thresholds on CRT FIFO. Fifo_aggressive gives more time to CPU for accessing the VRAM. Fifo_conservative reduces the possibility of noise caused when the CRT tries to read memory when it is used by CPU, but reduces performance. The default is between aggressive and moderate (more aggressive than moderate).

Information for SiS Users **:** *XF86Config Options*
Previous: *Supported chips*
Next: *Modelines*

4. Modelines

When constructing a modeline for use with the Sis driver you'll need to consider several points:

- H/W Acceleration. The H/W cursor, and fill operations currently allocate memory of the video ram for there own use. If this is not available these functions will automatically be disabled. Also, ext_eng_queue allocate 32k of Vram.
- Dot Clock. SiS documents the following video modes to work with 6326. The max dot clock allowable for your 6326 based board depends also on the memory installed on it. Option fast_vram can be needed for high dot clocks to work. Of course, the memory installed must allow 1 cycle R/W. The server tries to avoid problems with high dotclocks, limiting the maximum based on estimated memory bandwidth. Overriding the limits with dac_speed and modelines can damage the card if you exceed the card limits. Values between driver guess and chipset limits are acceptable, but can cause bad image quality, noise or no image displayed.
    - SiS recommended video modes for 6326:
    - 640x480 : 4, 8, 15, 16, 24 bpp at 85Hz Non-interlaced
    - 800x600 : 4, 8, 15, 16, 24 bpp at 85Hz Non-interlaced
    - 1024x768 : 4, 8, 15, 16, 24 bpp at 85Hz Non-interlaced
    - 1280x1024 : 4, 8, 15, 16, 24 bpp at 75Hz Non-interlaced
    - 1600x1200 : 4, 8 bpp at 65Hz Non-interlaced

5. Troubleshooting

The generic VGA driver doesn't work with 6326, so XF86Setup can't be used for this card. Please use xf86config instead.

With intensive generation there is a snow phenomenon on the screen. Can't remove it even if I used the fifo low/high water mark dumped from W95.

The latter point is changed. Now we use calculated values for the fifo settings, and this appears to be stable until the bandwidth required for CRT is near the memory bandwidth (dotclock*depth/8 near Mclk*32 or Mclk*64). In that case, you could try to use fifo_moderate, fifo_conservative or a lower dotclock.

Some video modes with high dot-clocks don't work at all, resulting on black screen. We are tracing now this problem. Lowering dotclock in that case could solve the problem.

```
Updated June 25, 1999 by Dirk Hohndel, covering changes for 530 and 620.
Updated October 12, 1998 by Juanjo Santamarta, covering changes for 5597 and 6326.
Updated November 6, 1998 by Juanjo Santamarta, covering changes for 5597, 86c2x5 and
6326.

$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/SiS.sgml,v 3.3.2.7 1999/06/25
08:57:14 hohndel Exp $
```

README.VIDEO7

TOC

Craig Struble

17 May 1994

1. The Driver:
2. Known bugs and What's been tested:
   2.1. Known bugs:
   2.2. What's been tested:
3. Who to contact:
4. Acknowledgments
   4.1. Thanks to:
   4.2. Other things I've already done:
   4.3. Things to do:
   4.4. Disclaimer:

1. The Driver:

The Video7 driver has only been tested on a Headland Technologies HT216-32 chip, but should work on other Video 7/Headland Technologies chips as well.

Currently this implementation of the video7 driver only supports single bank mode, which can cause performance degradation, and makes no attempt to distinguish between the different video7 chips.

It also does not probe for memory, so in your `XF86Config` file, make sure that you use the following line:

```
Videoram XXX
```

Where XXX is the amount of RAM in your card. Most of them have at least 512k, so this is a good value to start with.

Also, the clock probing function of XFree86 doesn't seem to correctly get the clocks. The documentation I used (vgadoc3) suggests using the following values for the `Clocks line` in your `XF86Config` file:

```
Clocks        25.175 28.322 30.000 32.514 34.000 36.000 38.000 40.000
```

For 800x600 mode, use a dot clock of 38 instead of 36 or 40 as suggested in most of the sample `XF86Config` files and `modeDB.txt`. This seems to be what is used in the BIOS mode (0x69) which is the 800x600 in 256 colors.

2. Known bugs and What's been tested:

2.1. Known bugs:

1. No video ram probing. Only known way to get this info is through an INT 10 call, but you can't do this in a user process.
2. Clock probing. I'm not sure the docs in vgadoc3 are correct.
3. Random lockups with the SVGA server

2.2. What's been tested:

1. An HT216-32 chip.
2. 800x600 mode and 640x480 mode
3. Mode switching and switching to text mode through `CTRL-ALT-F1`
4. Only been tested on Linux.

3. Who to contact:

Craig Struble (*cstruble@acm.vt.edu*) Video 7 driver

4. Acknowledgments

4.1. Thanks to:

- **Cara Cocking** for loving me and supporting me. Without her I'd be a bowl of jello.
- **XFree86 team** for the great stub code that allowed me to get this going.
- **Finn Thoegersen** for compiling vgadoc3. Without this I would not have had a clue.
- **Harm Hanemaayer** for the vga.c program in svgalib. Without this I would not have had the breakthroughs I needed to get the thing up and running.

4.2. Other things I've already done:

For Linux, I have a small patch to get the extended text modes to work on the Video 7 card.

4.3. Things to do:

- Try dual banking mode.
- Write an svgalib driver.
- Go back to graduate school. (I'm a glutton for punishment.)

4.4. Disclaimer:


```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Video7.sgml,v 3.5 1997/01/24
09:32:34 dawes Exp $
```

```
$XConsortium: Video7.sgml /main/3 1996/02/21 17:46:22 kaleb $
```

Information for Western Digital Chipset Users

The XFree86 Project, Inc.

14 July 1995

1. Supported chipsets

2. Special considerations

3. WD90C24 features

4. WD90C24 clocks

5. Additional WD90C24 information

TOC

1. Supported chipsets

XFree86 supports the following Western Digital SVGA chipsets: PVGA1, WD90C00, WD90C10, WD90C11, WD90C24, WD90C30, WD90C31, WD90C33. Note that the rest of the WD90C2x series of LCD-controller chipsets are still not supported. The WD90C24 family is now supported including acceleration, adjustable clocks and a full 1MB video ram even on dual scan systems (in CRT mode). If you have trouble with the new WD90C24 support (not that we expect you will), try specifying "wd90c30" or "wd90c31" on the `Chipset' line in your `XF86Config` file. The WD90C24, WD90C31 and WD90C33 are supported as an accelerated chipset in the SVGA server; the accelerated features are automatically activated when a WD90C24, WD90C31 or WD90C33 is detected, or specified in the `XF86Config` file.

2. Special considerations

All of the Western Digital chipsets after the PVGA1 support the ability to use the memory-refresh clock as an alternate dot-clock for video timing. Hence for all of these chipsets, the server will detect one more clocks than ``normal''. What this means is that if you have an old `Clocks' line in your XF86Config file, you should comment it out, and rerun the server with the `-probeonly' option to find all of the clock values. All but the last should be the same as what you had before; the last will be new.

For the WD90C00 chipset, the chipset will only support 640x480 in 256-color mode. Even though 512k of memory should allow better than 800x600, the chipset itself cannot do this. This is stated in the databook (which lists 1024x768x16 and 640x480x256 for specifications). We have also witnessed this behavior.

The server will detect 17 clocks for the WD90C24, WD90C30 and WD90C31 chipsets. If you have one of these chipsets, you should let the server re-probe the clocks and update your XF86Config.

There is an `Option' flag available for the XF86Config file that is specific to the Western Digital chipsets (except the WD90C24). This option is "swap_hibit". We have determined via experimentation that the WD90C1x and WD90C3x chipsets need the high-order clock-select bit inverted, and the PVGA1 and WD90C00 need it non-inverted. This is hardcoded into the driver. Since our sample-set was rather small, we have provided the "swap_hibit" option to invert this behavior. If the clocks detected by the server show a very low last clock (under 28Mhz), then this option is likely needed.

3. WD90C24 features

These next three sections apply only if you have a WD90C24, WD90C24a, or WD90C24a2 and don't specify some other chipset in your XF86Config file. The SVGA pvga1 driver now recognizes the wd90c24 family as different from the WD90C30 and seems to resolve most of the problems people encountered when these chips were treated as WD90C3X. The new code has the following features:

- Locks the shadow registers at appropriate times; This should prevent scrambled displays after exiting X with dual scan screens when simultaneous or LCD display mode is selected. The code does depend somewhat on the behavior of the BIOS regarding when it locks the shadow registers, etc.
- Allows (forces) the use of a full 1 Meg VRAM for dual scan systems when the server is started while external CRT only display is in operation. This allows 1024x768x8 resolution.
- If the `XF86Config` file specifies a virtual screen size which requires more than 512 K VRAM when the server is started on a Dual Scan LCD, the driver will force the virtual size to 640x480. This eliminates the need to edit the `XF86Config` file when you switch from 1024x resolution on the CRT, to or from the LCD screen. If no virtual size is specified, the result will be 800x600 virtual in LCD modes and 1024x768 in CRT only mode (so you have a choice).
- Note that on dual scan systems, you must still exit X, switch displays, and restart X to change to/from CRT only with 1 Meg videoram. This is because once the server starts, you can't change the virtual screen size. There is no way around this with the current server and the WD90C24 with dual scan displays. The WD90C24 requires half the videoram be used for a ``Frame buffer'' when the dual scan LCD is in use.
- The new server uses the accelerated features of the WD90C24a. It is not clear from the data book if the WD90C24 also supports ALL the required features. Several people have stated that the WD90C24 is not accelerated, but the differences section of the WD90c24a data book implies that they ARE all three accelerated. The differences documented with regard to acceleration are with the type of line drawing the hardware does; Only the newer chips support the type of line drawing that MS windows wants. This may be what has caused the confusion since the accelerated windows drivers may only support the WD90c24a chips. If this turns out to be a problem with the WD90C24, acceleration can be disabled by adding the line:

  ```
  Option "noaccel"
  ```

  to the Device section of the `XF86Config` file.
- Although the new server does not support programmable clocks in the same way as some of the other servers, 8 of the 17 clocks may be set to (almost) any value via the Clocks line. It also supports options for adjusting the VRAM clock.

4. WD90C24 clocks

Here are some more details on the adjustable clocks:

The VRAM clock (Mclk) is adjusted by adding ONE of the following option lines to the Device section of the `XF86Config`:

```
        Option      "slow_dram"     # Set Mclk to 47.429 MHz
        Option      "med_dram"      # Set Mclk to 49.219 MHz
        Option      "fast_dram"     # Set Mclk to 55.035 MHz
```

The default is to leave Mclk as the BIOS sets it. This is 44.297 on many systems. Some systems may not work properly with any of these options. If you experience ``bit errors'' on your display, reduce the Mclk speed, or don't use any of these options. The Mclk is not reset on server exit.

The data book says that the maximum pixel clock is 1.6 times Mclk so you may want to experiment with higher Mclk rates if you have a fast monitor. It also says a 44.297MHz Mclk and 65MHz pixel clock is the fastest the WD90C24A2 is designed to go. However, some success has been reported with faster clocks. Don't expect all the clocks the chip can provide to work properly.

The second and fourth group of 4 clocks are adjustable. That is, clocks 5, 6, 7, 8 and 13, 14, 15, 16 (counting from 1). These clocks are set by the Clocks line. Be sure to adjust the 17th (last) clock to match your Mclk. Here is a sample set of clocks lines with some clocks defined which are not directly provided by the chip. The NON-programmable clocks (1-4 and 9-12) MUST be set as indicated here.

```
    Clocks      25.175 28.322 65     36     # These are *not* programmable

    Clocks      29.979 77.408 62.195 59.957 # these are programmable
    Clocks      31.5   35.501 75.166 50.114 # these are *not* programmable
    Clocks      39.822 72.038 44.744 80.092 # these are programmable
    Clocks      44.297                     # Change this if you change
                                           #   Mclk above.
```

You can program the clocks in increments of .447443 MHz. The server will warn you and adjust to the nearest increment if you specify a clock which does not fit this formula. Clocks 1-4 and 9-12 (the fixed clocks) are not constrained to this multiple, but instead are used to provide standard clocks which are not a multiple by .447443 MHz.

If you probe for clocks (for example to find your Mclk), do it in CRT only mode and then add clocks lines in your `XF86Config` file. Clocks will not probe correctly in LCD mode on most systems.

The BIOS on some systems may not allow switching from CRT to LCD unless the correct clock and/or mode is used. Try the following mode line for 640x480 LCD displays.

```
    ModeLine "640x480"  25.175   640 664 760 800     480 491 493 525 #CRT/LCD
```

The following modelines have been tested with the above Clocks lines on some systems, and are provided here as examples. Some testers have experienced minor problems (snow) with the fixed 65 and 75.166 MHz dot clocks. The modelines below have been reported to circumvent these problems. Do not assume your monitor will not be damaged by any of these.


```
 # VESA 800x600@72Hz Non-Interlaced mode
   ModeLine  "800x600.50"  50  800  856  976 1040 600 637 643 666  +hsync +vsync


 # 1024x768  Interlaced mode
   ModeLine  "1024x768i"   45  1024 1048 1208 1264 768 776 784 817 +hsync +vsync
Interlace

 # 1024x768@60Hz Non-interlaced Mode
 # One of the dram options may be necessary
   ModeLine  "1024x768.65" 65  1024 1032 1176 1344 768 771 777 806 -hsync -vsync


 # 1024x768@60Hz Non-Interlaced mode (non-standard dot-clock)
 # Seems to work without dram options
   ModeLine  "1024x768.62" 62  1024 1064 1240 1280 768 774 776 808

 # 1024x768@70Hz Non-Interlaced mode (non-standard dot-clock)
 # May need fast_dram option
   ModeLine  "1024x768.72" 72  1024 1056 1192 1280 768 770 776 806 -hsync -vsync
```

Information for Western Digital Chipset Users : *WD90C24 clocks*

Previous: *WD90C24 features*

Next: *Additional WD90C24 information*

5. Additional WD90C24 information

Standard disclaimers apply. Use this driver at your own risk. If you need additional information on using XFree86 with the WD90C24 family however, you might try Darin Ernst's home page. Darin maintains a mini-HOWTO on ``X and the WD90C24". He was the first tester of the WD90C24 code and provided many good ideas and encouragement. You can reach Darin at *darin@castle.net* or *dernst@pppl.gov*. I only provided the WD90C24 specific code. You can reach me (Brad Bosch) at *brad@Lachman.com*.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/WstDig.sgml,v 3.6 1997/01/24
09:32:38 dawes Exp $
```

```
$XConsortium: WstDig.sgml /main/5 1996/02/21 17:46:29 kaleb $
```

Information for W32 and ET6000 Chipset Users

Glenn G. Lai *<glenn@cs.utexas.edu>*, Dirk H. Hohndel *<hohndel@XFree86.Org>*, Koen Gadeyne *<koen.gadeyne@barco.com>*

May 16, 1997

1. Information for W32 Chipset Users
2. Using XF86_W32 on a board with an ICS5341 GENDAC
3. Using XF86_W32 on a board with an STG1703 GENDAC
4. Using XF86_W32 on an ET6000-based board
5. Using XF86_SVGA with ET4000/W32 and ET6000 cards
6. Acknowledgments

1. Information for W32 Chipset Users

XF86_W32 gets phased out, now that the SVGA server with XAA acceleration is at least as fast as the W32 server but supports more cards and for some even higher color depths. For details about using the XF86_SVGA with W32 cards, look below. Note that currently not all cards that are accelerated by XF86_W32 are accelerated by XF86_SVGA at this moment (only ET6000 and ET4000W32p to be exact).

XF86_W32 is supposed to be the stable server for cards that worked before and have trouble with the new XF86_SVGA. Use this server when the SVGA server fails to work for you (this happens on some ET4000W32 ISA cards), or when it refuses to accelerate anything (on ET4000W32i for example).

Since XFree 3.2A, this server has not been updated. This means that some (known) bugs have not been fixed. They are fixed in the SVGA Tseng driver (or replaced by others...), so if you have problems, try the SVGA server instead.

XF86_W32 is basically XF86_SVGA with the drawing code completely replaced with one based on X11R6's mi/cfb code and modified for the ET4000/W32 series. Even though it accepts the same keywords as XF86_SVGA, those not applicable to the ET4000/W32 series are silently ignored; e.g., the keyword "SpeedUp" is a no-op. The server currently supports the w32, w32i, w32p and et6000 chips. For a complete list, see the sign-on message printed by XF86_W32. The server only supports 256 colors.

Just as with XF86_SVGA, you can specify a virtual world that has a width that is a multiple of four. The size of the virtual world is constrained by the amount of the available video RAM. XF86_W32 can use more than 1 M of video RAM, but it reserves 1 K for internal use. If you have 1 M, XF86_W32 claims you have 1023 K; you get to specify the virtual world as 1152x900, but not 1152x910.

For most cards the maximum clock is set to 86 MHz according to the Tseng databooks. For a non-interlaced 1280x1024x(256 colors) at say 135-MHz, you need a w32p (with its 16-bit RAMDAC bus) with a multiplexing RAMDAC so that the w32p sees only (135/2 = 67.5) MHz, not 135 MHz. This requires special code only provided for cards using the ICS5341 GENDAC or the STG1703. This code seems to work fine for most people, except, with the ICS5341, for a small band of frequencies around 90MHz.

If you have problems with the server. Try the following:

- Put Option "slow_dram" in the Device Section.
- Put Option "pci_burst_off" in the Device Section.
- Put Option "w32_interleave_off" in the Device Section.
- Take out the Hercules monochrome adapter, if you have one. Many configurations of the ET4000/W32 series do not allow one in the system.
- Get a motherboard with its local bus running at 33 MHz. Many, if not all, ET4000/W32 boards

will surely behave in a funny way on a 50-MHz bus. You may have to use a wait state or two, first try without any.

- Cold-boot your machine. Do not run anything that messes with the video hardware, including other X servers, before running `XF86_W32`.
- In case of an ET6000 card, try specifying chipset "et6000" in the Device Section. The card normally auto-probes from the PCI bus, but on some systems, another on-board VGA card, although disabled, may cause the ET6000 server to want to use the other card.
- Try XF86_SVGA. If it works, put the following in your XF86Config:

```
Ramdac "generic"
```

Note that the built-in power saver (for a "green" monitor) has not been tested. Also do not expect it to work on a board without a w32p_rev_c or later chip. This option is currently disabled completely, because it causes video memory corruption (or even a crash). The SVGA server (`XF86_SVGA`) supports VESA DPMS, and doesn't corrupt the screen.

Information for W32 and ET6000 Chipset Users **:** *Information for W32 Chipset Users*

Previous: *Information for W32 and ET6000 Chipset Users*

Next: *Using XF86_W32 on a board with an ICS5341 GENDAC*

2. Using XF86_W32 on a board with an ICS5341 GENDAC

Even though the GENDAC provides a set of standard clocks that can be found with the normal clock probing procedure, it is mandatory to put a

```
    ClockChip  "ics5341"
```

line into the Device Section to be able to use the programmable clocks that the ICS5341 can produce. You can also add a

```
    Ramdac     "ics5341"
```

line, but the RAMDAC should be auto-probed correctly. Even though the server currently accepts any dot clock up to 135MHz with the ICS5341 GENDAC, most boards show a small band of clock values in the area between 86MHz and about 100MHz that don't work. This are usually is just a few MHz wide, higher clocks as well as lower clocks work just fine. I'm working on it. (DHH)

TOC

3. Using XF86_W32 on a board with an STG1703 GENDAC

Even though the STG1703 provides a set of standard clocks that can be found with the normal clock probing procedure, it is mandatory to put a

```
    ClockChip  "stg1703"
```

line into the Device Section to be able to use the programmable clocks that the STG1703 can produce. You can also add a

```
    Ramdac     "stg1703"
```

line, but the RAMDAC should be auto-probed correctly.

4. Using XF86_W32 on an ET6000-based board

The ET6000 driver code was developed on top of the existing ET4000/W32 code, because of the many similarities between both devices. As with the other W32 (external) clockchip/RAMDAC devices, the ET6000's built-in clockchip/RAMDAC provides a set of 8 standard clocks, which could be probed with the normal XFree clock probing procedure. In spite of that, XF86_W32 will always use the built-in programmable clockchip and RAMDAC. So there is no need for a

```
ClockChip "et6000"
```

or a

```
Ramdac "et6000"
```

line in the Device Section of the XF86Config file. Once it knows it's dealing with an ET6000, XF86_W32 will find its own way. At this moment, accelerated support is very sketchy, and only uses those things the ET4000/W32 code already provided, with some changes due to incompatibilities between the two devices. Major speed improvements should be possible. Tseng Labs specifies a maximum pixel clock of 135 MHz for the ET6000 chips (with higher clocks to come).

There is a known bug in this server when using it with ET6000 cards with 2.25 MB MDRAM: the server will detect 2.5 MB instead, and as a result, most accelerated operations won't work. On cards with 2.25 MB MDRAM, you *must* insert a

```
VideoRam 2304
```

line in your XF86Config.

5. Using XF86_SVGA with ET4000/W32 and ET6000 cards

Starting with XFree86-3.2A, the SVGA server uses the new XFree86 Acceleration Architecture (XAA). With this technology XF86_SVGA is at least as fast if not faster than the XF86_W32 with the same hardware. Additionally, it supports higher color depths with some cards. On the downside, some special RAMDACs and clock chips that are supported in XF86_W32 for W32 cards are not supported in the SVGA server at this point.

If the SVGA server does not give a picture with your W32 card try the following:

- Put Option "slow_dram" in the Device Section.
- Put Option "pci_burst_off" in the Device Section.
- Put Option "w32_interleave_off" in the Device Section.
- Put Option "no_accel" in the Device Section.
- Cold-boot your machine. Sometimes it is even necessary to physically turn of the power for the W32 chip to get in a sane state again. Do not run anything that messes with the video hardware, including other X servers, before running XF86_SVGA.

6. Acknowledgments

Jerry J. Shekhel (*jerry@msi.com*) gave me (GGL) the 1-M Mirage ET4000/W32 VLB board on which the initial development (X_W32) was done.

X11R6 and The XFree86 Project provide the base code for XF86_W32.

Hercules Computer Technology Inc. lent me (GGL) a 2-M Hercules Dynamite Pro VLB board for the development that led to XF86_W32. They donated a Dynamite Power PCI to The XFree86 Project, that was used by DHH to extend the server.

Koen Gadeyne (*koen.gadeyne@barco.com*) wrote a patchkit for XFree86-3.1.1 that was partly integrated in this server and he continues to help develop it.

Tseng Labs Europe kindly donated (KMG) an ET6000-based board (a Jazz Multimedia G-Force 128), which spurred the development of the ET6000 code.

Numerous testers have given me feedback for X_W32 and later XF86_W32. I apologize for my failure to keep track of the people who tested X_W32, but the names of the people involved with the XF86_W32 testing are listed below:

Linux:

bf11620@coewl.cen.uiuc.edu (Byron Thomas Faber)
dlj0@chern.math.lehigh.edu (David Johnson)
peterc@a3.ph.man.ac.uk (Peter Chang)
dmm0t@rincewind.mech.virginia.edu (David Meyer)
nrh@philabs.Philips.COM (Nikolaus R. Haus)
jdooley@dbp.caltech.edu (James Dooley)
thumper@hitchcock.eng.uiowa.edu (Timothy Paul Schlie)
klatta@pkdla5.syntex.com (Ken Latta)
robinson@cnj.digex.net (Andrew Robinson)
reggie@phys.washington.edu (Reginald S. Perry)
sjm@cs.tut.fi (M{kinen Sami J)
engel@yacc.central.de (C. Engelmann) **use** *cengelm@gwdg.de*
postgate@cafe.net (Richard Postgate)
are1@cec.wustl.edu (Andy Ellsworth)
bill@celtech.com (Bill Foster)

FreeBSD:

ljo@ljo-slip.DIALIN.CWRU.Edu (L Jonas Olsson)

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/W32.sgml,v 3.16.2.4 1997/06/01
12:33:36 dawes Exp $
```

```
$XConsortium: W32.sgml /main/11 1996/10/19 18:03:45 kaleb $
```

TOC

Information for Tseng Chipset Users

The XFree86 Project, Inc. Dirk H. Hohndel, Koen Gadeyne and others.

03 Nov 1998

1. Supported chipsets
2. Terminology
3. ET4000 driver features
4. ET6000 driver features
5. Clock selection problems with some ET4000 boards
6. Text mode restore problems
7. Basic configuration
8. general options in the `XF86Config` file
9. linear memory base address (MemBase) issues
    - 9.1. What you should know BEFORE trying another MemBase
    - 9.2. Choosing a MemBase
    - 9.3. An alternative approach
    - 9.4. When all else fails...
    - 9.5. Restrictions
    - 9.6. Some boards simply cannot work in linear mode
    - 9.7. How can I see if the linear address is wrong?
10. Mode issues

TOC

11. Acceleration issues

12. ET6000 memory size facts and fiction

13. ET6000 memory bandwidth hype and the impact on video modes

14. Linear addressing and 16bpp/24bpp/32bpp modes

15. Trouble shooting with the SVGA Tseng driver

16. Acknowledgments

1. Supported chipsets

The Tseng chipsets supported by XFree86 are ET3000, ET4000, ET4000/W32 and ET6000. Accelerated features of the ET4000/W32, W32i, W32p and ET6000 are supported by the SVGA driver. For details about the separate accelerated 8bpp (=256 color) ET4000/W32 and ET6000 server, refer to README.W32.

Note that you should NOT be using XF86_W32 unless XF86_SVGA doesn't work on your hardware. No further development is being done on the W32 server; all new efforts go into the SVGA server.

Some ET4000W32 ISA cards are known NOT to work with the SVGA server in this version (XFree86 3.3.1): they hang the machine... Use the W32 server `XF86_W32` for these cards!

2. Terminology

In the rest of this document, "8bpp" is short for "8 bits per pixel", which means a 256-color mode. Similarly, 15bpp refers to 32768 colors, 16bpp to 65536 colors , 24bpp to a "packed" 16 million color mode, and 32bpp to a "sparse" 16 million color mode (at 32bpp, only 24 of the 32 bits are actually used, hence the "sparse").

15bpp is only used here to differentiate it from 16bpp, but they are both normally referred to as 16bpp. 15bpp is actually 16bpp with a 5-5-5 color weight (wasting one bit per pixel), while 16bpp is, well, 16bpp, with 5-6-5 color weight.

3. ET4000 driver features

The SVGA driver for ET4000 chipsets supports all color depths (8, 15, 16, 24 and 24 bpp) on most ET4000 chips starting with the ET4000W32i. The ET4000W32 only supports 8bpp. Depending on the RAMDAC and the support code in the SVGA server, some cards may only support a few of these color depths, or even only 8bpp.

On W32i and W32p chips all color depths are supported on the supported RAMDACs (currently ICS5341, STG170x and Chrontel CH8398). These modes are also accelerated.

Some W32p board implementations are limited to 1 MB of video memory in linear memory modes. This is a hardware limitation that cannot be solved in the driver. Since XFree86 requires linear memory for 16/24/32 bpp modes, the usefulness of these cards for highcolor and truecolor applications is severely limited (those modes mostly use a lot of video memory).

In addition, those cards also don't support acceleration in linear mode. This is a design choice in the driver code: if acceleration were to be supported in linear mode, you'd only be able to use 768 kb of video memory, and the driver code would be twice as complex.

Cards with a RAMDAC that is not yet supported will be limited in a similar manner as the older cards, i.e. to a maximum pixel clock of 86 MHz, whilst they actually might be able to go up to 135 MHz. As a result, 1280x1024 modes will only be possible when using interlacing, and non-interlaced modes are limited to about 1024x768 at 75 Hz refresh.

For a non-interlaced 1280x1024x(256 colors) at say 135-MHz on a W32-type card, you need a w32p (with its 16-bit RAMDAC bus) with a multiplexing RAMDAC so that the w32p sees only (135/2 = 67.5) MHz, not 135 MHz. This requires special code only provided for cards using the ICS5341 GENDAC, the STG170x or the CH8398. This code seems to work fine for most people, except, with the ICS5341, for a small band of frequencies around 90MHz.

Linear memory mode (especially important for some DGA clients, like xf86quake) is supported on all ET4000W32i and ET4000W32p cards, but not on the ET4000W32. See the section on linear memory for more information. There are some important issues related to linear memory.

For the higher color depths (16, 24 and 32 bpp), linear memory mode is REQUIRED. It is enabled by default in these modes. There is no need to specify that in the `XF86Config` file. Please read the section on linear memory below: it contains some vital information on how to avoid serious problems.

To force "banked" mode in 8bpp modes (where linear memory mode is the default), put the following in the Device section of your `XF86Config`:

```
  Option "no_linear"
```

Acceleration support is present, and enabled by default, for all W32 and ET6000 family chips. This i based on the new XFree86 acceleration interface (XAA).

If you have problems with acceleration, acceleration can be disabled by putting the following in the Device section of your `XF86Config`:

```
Option "noaccel"
```

On some PCI systems (i.e. only on the ET6000 and the ET4000W32p), acceleration may cause occasional font corruption. This is probably caused by a badly written system BIOS that ignores the fact that the Tseng PCI devices have their "non-prefetchable" attribute set. On such a BIOS, a PCI feature called "write combining" (or "byte merging") is enabled for the Tseng video card, although it is not permitted. Some systems allow you to manually enable or disable the Write Combining feature in the BIOS setup (sometimes abbreviated to WC). Make sure WC is disabled for the VGA memory aperture.

If you experience font corruption on your system and are unable to manually disable WC in your BIOS, font acceleration may be disabled using the following in the Device section of your `XF86Config`:

```
Option "xaa_no_color_exp"
```

Note that this will reduce the performance of the X server.

Information for Tseng Chipset Users : *ET4000 driver features*
Previous: *Terminology*
Next: *ET6000 driver features*

4. ET6000 driver features

In addition to the features in the ET4000 driver, the SVGA ET6000 server supports all possible color depths in the SVGA server: 8bpp, 16bpp (both at 5-5-5 and 5-6-5 color resolutions), 24bpp and 32 bpp.

Linear memory mode (as opposed to the VGA default, banked memory layout) is supported. It is required and enabled by default for the 16/24/32 bpp modes. For 8bpp, the default is linear mode for PCI cards and banked mode for ISA/VLB cards.

To force linear memory at 8bpp, put the following in the SVGA section of your `XF86Config`:

```
Option "linear"
```

Acceleration is supported and is enabled by default, and accelerates all color depths on the ET6000. Acceleration can be disabled by adding the following in the Device section of your `XF86Config`:

```
Option "noaccel"
```

The hardware cursor is supported in all color depths. Due to a hardware limitation in the ET6000, only a limited set of colors is supported (2 significant bits per color component). This may cause some (small) cursor color errors. If absolute cursor color accuracy is required, the hardware cursor should not be enabled. However, in most applications, this will not be a problem. The hardware cursor can be enabled using

```
Option "hw_cursor"
```

There is a problem with the hardware cursor at high dotclocks (above approx. 110MHz) at which point the cursor does strange things when partly off the left-hand side of the screen.

On older ET6000 chip revisions, DoubleScan modes currently don't work with the hardware cursor: only the top half of the cursor is visible. If you want to use DoubleScan modes (320x200 is a popular one), then do not enable the hardware cursor. Most recent ET6000 cards and the ET6100 do not exhibit this problem.

On some PCI systems, acceleration may cause occasional font corruption. As described above, this is caused by a bug in your system BIOS or a wrong setting of the write combining feature in that BIOS. If you are unable to fix the BIOS or force the option off, font acceleration may be disabled using the following in the Device section of your `XF86Config`:

```
Option "xaa_no_color_exp"
```

When using accelerated high color-depths (24bpp and 32bpp), high-resolution modes (starting

somewhere around 800x600) may cause temporary "garbage" lines to the right of the screen while t accelerator is busy. The garbage should not be persistent: it should go away as soon as the server is l alone. This is a memory bandwidth problem, and thus cannot be resolved (except by not allowing such modes at all, which is what is done in the current driver).

Ignoring it is one option (it isn't destructive). Disabling acceleration in the Device section of the `XF86Config` file is another option: since the accelerator is not being used, there is ample bandwidth to avoid such problems.

Information for Tseng Chipset Users **:** *ET6000 driver features*
Previous: *ET4000 driver features*
Next: *Clock selection problems with some ET4000 boards*

5. Clock selection problems with some ET4000 boards

XFree86 has some problems getting the clock selection right with some ET4000 boards when the server is started from a high-resolution text mode. The clock selection is always correct when the server is started from a standard 80x25 text mode.

This problem is indicated when the reported clocks are different when the server is started from the high-resolution text mode from what they are when it is started from the 80x25 text mode. To allow the server to work correctly from the high-resolution text mode, there are some Option flags that may be set in `XF86Config`. To find out which flags to set, start the server with the -probeonly flag from an 80x25 text mode and look at the information printed by the server. If the line:

```
VGAXXX: ET4000: Initial hibit state: low
```

is printed, put the following in the SVGA, VGA16 and VGA2 sections of your `XF86Config`:

```
Option "hibit_low"
```

If the line:

```
VGAXXX: ET4000: Initial hibit state: high
```

is printed, put the following in the SVGA, VGA16 and VGA2 sections of your `XF86Config`:

```
Option "hibit_high"
```

6. Text mode restore problems

In XFree86 1.3, an option flag ``force_bits'' was provided as an experiment to attempt to alleviate text-restoration problems that some people experienced. We have now made the behavior of this option the default, hence the flag has been removed. Hopefully the past text-restoration problems are alleviated in XFree86 2.0.

TOC

7. Basic configuration

It is recommended that you generate an XF86Config file using the `XF86Setup`' or `xf86config`' program, which should produce a working high-resolution 8bpp configuration. You may want to include mode timings in the `Monitor` section that better fit your monitor (e.g 1152x864 modes). The driver options are described in detail in the next section; here the basic options are hinted at.

If graphics redrawing goes wrong on accelerated chips (ET4000W32 and ET6000), first try the `"noaccel"` option, which disables all accelerated functions.

8. general options in the `XF86Config` file

The following options are of particular interest to the Tseng driver. Each of them must be specified in the `svga' driver section of the `XF86Config` file, within the `Screen` subsections of the depths to which they are applicable (you can enable options for all depths by specifying them in the Device section).

Option "noaccel"

(ET4000W32p, et6000) This option will disable the use of any accelerated functions. This is likely to help with some problems related to DRAM timing, high dot clocks, and bugs in accelerated functions, at the cost of performance (which will still be reasonable on a local or PCI bus). This option applies only to those chips where acceleration is supported.

Option "fast_dram" "slow_dram"

These options set the DRAM speed of certain cards where it applies.

The `"slow_dram"` option is always enabled on ET4000, and ET4000W32. If enabled, it slows down DRAM timing, which may avoid some memory-related problems. If your card starts up with a black screen (and possibly a system hang), this option might be needed.

The `"fast_dram"` option will cause the driver to speed up DRAM timings, which may also avoid screen-related problems (streaking, stripes, garbage, ...). It may also increase those very same effects.

All in all, these are potentially dangerous options: they could crash your machine as soon as you start the server. Use them with caution.

option "w32_interleave_off" "w32_interleave_on" (W32i, W32p)

Force memory interleaving off or on. W32i and W32p chips can increase memory bandwidth when they have 2MB or more video memory. Normally the VGA BIOS sets the W32i or W32p chip to the correct mode. If you suspect problems with memory sizing or interleaving, fooling around with these options may improve the situation. It may also make things worse. These options are not normally needed: the server will use the correct value automatically. Setting this option the wrong way will result in a completely distorted display.

option "pci_burst_off" "pci_burst_on" (W32p)

This option disables or enables PCI bursts on the W32p chip if it's a PCI card. Normally, a good BIOS will set the motherboard and the VGA card to the same setting, but if both don't match, you may experience garbage on the screen (e.g. mouse droppings). These options allow you to match the W32p burst setting to the motherboard setting.

videoram 1024 (or another value) (all chips)

This option will override the detected amount of video memory, and pretend the given amount of

memory is present on the card. This is useful on cards with 2Mbyte of memory whose DRAM configuration is not compatible with the way the driver enables the upper megabyte of memory, or if the memory detection goes wrong. It must be specified in the Device section.

Clockchip "et6000" (et6000)

This enables programmable clocks, but obviously only on the et6000. It must be specified in the Device section. Normally the server will automatically use this feature when it detects an ET6000. Use it only when you suspect auto-detection is not working. Note that some frequencies may be unstable (resulting in a `Wavy' screen). Only tried and tested frequencies (like the default clocks) are guaranteed to be stable. If this happens, try a slightly different frequency in the modeline (like 0.5 MHz more or less). The monitor should still be capable of syncing to this frequency, but the clockchip may already be outside an unstable region.

Option "linear" (ET4000W32i, ET4000W32p, ET6000)

This enables linear addressing, which is the mapping of the entire framebuffer to a high address beyond system memory, making the full length of the framebuffer directly accessible. In this way, slow SVGA bank switching (where only a small fraction of the framebuffer is visible at one time) is not necessary. It enhances performance at 256 colors, and is currently required for 16bpp, 24bpp, and 32bpp.

MemBase 0xE0000000. (or a different address) (ET4000W32, ET6000)

This sets the physical memory base address of the linear framebuffer. It must be specified in the Device section. It may be required for non-PCI linear addressing configurations, and might be useful for PCI-based systems where auto-detection fails. However, almost all PCI systems will not need this.

Read the section on linear memory base address issues below!

Read the section on linear memory base address issues below! (Message repeated for a very good reason)

Use this option ONLY if you have trouble with the default MemBase used by the server, or if the server explicitly states that you must provide one.

Option "pci_retry" (ET4000W32p on PCI bus, ET6000)

This enables the PCI bus retry function, which is a performance enhancing mode for local bus or PCI bus-based systems, where the VGA controller will put the bus in a hold state (sort of like wait-states) when the server tries to start a new accelerated operation but the accelerator is still busy with the previous operation.

This is the fastest way to drive a VGA card (no busy-waiting loops needed), but it also stresses some hardware that is timing-dependent (tape drives, sound cards, etc). See also the trouble shooting section.

---

9. linear memory base address (MemBase) issues

First a WARNING: defining a bad MemBase may cause serious injury or death (to your operating system, of course). Especially defining the MemBase to be inside the range of system memory is a ticket to hell.

9.1. What you should know BEFORE trying another MemBase

Rule #1: first, let the server find a memory base by itself, without specifying it. Make sure you "sync" all files to disk and close all critical applications. Make sure nothing bad will happen to your filesystems if you have to jump for the power switch soon.

The most critical cards are the ET4000W32p rev a and rev b on VESA local bus (VLB). The server will autodetect a linear base address that doesn't work on all systems.

The least critical cards are PCI-bus cards. The PCI BIOS normally takes care of assigning a good MemBase, and you should never have to deal with all the mumbo-jumbo below.

If the server gets it wrong, you may end up with a severe system crash (e.g. if it maps the video memory right on top of your system memory). If this happens, RESET IMMEDIATELY. Do not try to shut down cleanly, because the X-server, thinking it writes to the VGA memory, will write to system memory instead, and you'll be writing corrupted data to disk. If you did a sync prior to starting the server, there will be no harm done (only a filesystem check which should end up clean). DO NOT attempt to redirect the server output to a file on the system you're testing on (that will write data after you synced).

These are worst-case scenarios, and it is very unlikely this will happen to you. The text above is to make sure you are properly prepared, so that nothing serious happens.

When the server can't find a working linear memory base, it's time to experiment. The rest of this section deals with that.

9.2. Choosing a MemBase

Choosing a suitable MemBase can be quite tricky. If you have no way of determining the MemBase your card uses, trying to put it a few Mb above the system memory is a good first guess. E.g. if you have 16 Mb of RAM, defining MemBase 0x01000000 (=16M) or 0x01400000 (=20M) may work.

However, this may only work on non-PCI systems, as PCI systems mostly map all hardware above the 2GB mark. But then again, on PCI systems the server is almost always able to detect the correct linear memory base address. The only exception are those systems with more than one PCI VGA card.

On most VESA local bus (VLB) boards, there is an additional problem with address decoding. Most

motherboards only decode the first 32, 64 or 128 MB of address space (a good pointer is to check the amount of DRAM that can be installed on the board: it will at least decode as much address space as it supports DRAM).

On such boards, you MUST specify a MemBase inside that range, or the actual address may wrap back onto system memory: if your system only decodes 128MB of addresses, and you set the MemBase to 128 MB, it will actually be decoded as being on address 0, which is probably exactly where your kernel memory is located. That is why the general guideline of putting the MemBase just above the system memory is a sound one: it stands most chance of actually being inside the decoded address range of the board. Unless your motherboard's entire memory space is filled with RAM.

9.3. An alternative approach

If you don't know how much memory address space your motherboard decodes (and who does?), try using a "non-trivial" address, like 0x1FC00000, which has enough bits set to "1" to work on any motherboard, even if a few are not decoded. Keep in mind that using for example 0x10000000 may end up right on top of your system memory if the motherboard doesn't decode all upper address bits. You will only do that once.

9.4. When all else fails...

Some other VLB boards can only map the linear framebuffer above the 1GB mark (0x80000000 and up), so you must use a MemBase that is higher or equal to 0x80000000.

Some other VLB boards can only map the linear framebuffer BELOW the 16 MB mark. So you may want to try booting your system with up to 12 MB of memory (some operating systems allow you to supply a boot-time parameter that limits the memory to a certain amount, so you don't have to open your computer to try this), and set the MemBase to 0x00C00000 (=12M).

Unfortunately, there is no easy way to tell what system you have (these details are mostly not in the motherboard manuals). Trial and error is the only road to success here. The server code will provide a default that works on most boards... but yours won't be one of those, of course.

9.5. Restrictions

There are some limits as to where the linear memory base may be put. On any ET4000W32, it must have a 4MB granularity (i.e. it can be put at 16M or at 20M, but not at 18M). On ET6000, it needs a 16M granularity (note: the ET6000 driver should be able to determine the linear memory base automatically, so you should never need to define MemBase in the first place).

On ET4000W32i, things are worse: the linear address base is hardwired on the card, and there is no reliable way to read it back from the card. You need to know the address in some way, and specify it. The current code does an intelligent guess at it, but this is no guarantee.

On ISA cards, things are much more simple: ISA only uses 24 address lines, and hence the linear memory MUST lie within the 16 MB boundary. Together with the 4MB granularity of the linear memory base address on ET4000 cards, this means that you cannot have more than 12 MB of system memory in

the machine if you want to use linear memory. Hence, the only realistic MemBase for ISA cards is 0x00C00000. This is also what the server will automatically choose if it detects an ISA W32 card.

WARNING: you must not have over 12 MB of system memory in this case. Or if you have it, you must disable access to all memory above the 12 MB mark. Some operating systems allow you to specify at startup how much memory it is allowed to use, so you don't have to unplug some memory each time you want to use linear memory.

9.6. Some boards simply cannot work in linear mode

Yes, and in that case, you're out of luck.

There can be at least two reasons for this.

The first is the most common: the board manufacturer has left out the necessary connections and hardware to be able to use linear addressing. This means that no coding effort on this planet can help you with your problem: it is physically impossible to use linear addressing.

The second reason is that the current XFree86 Tseng linear addressing code is incompatible with the way your board is designed. The XFree86 Tseng code assumes a 1:1 mapping of the address lines from the bus (either ISA, VLB or PCI) to the address lines on the Tseng VGA chip. As unlikely as it may sound, this may NOT be the case!

Some very rare boards do not have such a 1:1 mapping (e.g. two address lines swapped). It is possible to support this type of hardware, but at this moment, this has not been implemented yet.

Other boards use external address decoding hardware that combines a number of address lines on the bus to a (smaller) number of address lines to the VGA chip. One such board for example uses three NOR gates (one 74F02 chip) to combine the 6 upper address lines to three address pins on the W32i chip. Obviously, this represents a 2:1 mapping, and not a 1:1 mapping. Therefor, this board is not "compatible" with the way XFree86 implements linear mode.

9.7. How can I see if the linear address is wrong?

Simple: nothing works, or your machine locks solid, or it crashes, or a zillion of other things.

However, sometimes it is not always as obvious. Sometimes nothing bad happens: you just get a black screen, or a screen with rubbish on it, but nothing is drawn on it. Sometimes you get a core dump when the first application starts.

If acceleration is enabled in those cases, you will almost always see multiple "WAIT_ACL: timeout" messages in the server output. That is because the accelerator registers are also mapped in the linear memory, and if linear memory doesn't work, then also the accelerator doesn't work.

NOTE however that a WAIT_ACL message doesn't necessarily mean the linear memory address is bad. There are a number of other reasons for this message as well. But if you never saw these messages at 8bpp banked, then there's a good chance you have a linear memory problem ("banked" is the opposite of "linear", and is the default mode when "option linear" is not in the XF86Config file).

TOC

10. Mode issues

The accelerated driver on ET4000W32/W32i/W32p and ET6000 needs at least 1K bytes of scratch space in video memory. Consequently, if you want acceleration, a 1024x1024 virtual resolution should not be used with a 1Mbyte card. This also means that a 1024x768 mode at 24bpp on a 2.25 MB ET6000 card cannot be accelerated, since you've used up all the memory for the display.

The same thing goes for the ET6000 hardware cursor: it also requires 1kb of free video memory. If that memory is not available, the hardware cursor cannot be used.

The use of a higher dot clock frequencies has a negative effect on the performance of graphics operations on non-et6000 cards (the effect is much less, or even non-existing, on ET6000 cards), especially BitBlt, when little video memory bandwidth is left for drawing. Memory bandwidth is the speed at which data can be pumped into the memory while the RAMDAC is pulling it out to display it on the screen.

Higher dot-clocks (mostly related to higher resolutions) consume more bandwidth, so that less of it is left for drawing into the framebuffer. With a working accelerator, things become increasingly crammed, because modern accelerators can consume huge amounts of bandwidth (but they also give you high speeds in return). High color depths also need extra bandwidth.

If you are short on memory bandwidth (see the separate section on this) and experience blitting slowness or screen "glitches", try using the lowest dot clock that is acceptable; for example, on a 14" or 15" screen 800x600 with high refresh (50 MHz dot clock) is not so bad, with a large virtual screen.

Tseng chips are mostly known for their (very) good memory bandwidth, so you should only start to see problems in the higher regions.

It does not make much sense performance-wise to use the highest clock (85 MHz) for 1024x768 at 76 Hz on a 1 MB ET4000W32; the card will very slow, because there is almost no bandwidth left for drawing. A 75 MHz dot clock results in 70 Hz which should be acceptable. If you have a monitor that supports 1024x768 at 76 Hz with a 85 MHz dot clock, an 1MB card is a poor match anyway.

The ET4000W32i and ET4000W32p have a special feature that almost doubles memory bandwidth (+70%) using "interleaving" between the two banks. Upgrading to 2MB is a real bonus on these cards. This is not true for W32 cards or for ET6000 cards.

11. Acceleration issues

The XFree Acceleration Architecture makes extensive use of the unused video memory on the VGA card. If there is not enough free video memory, some acceleration features will be disabled or crippled, resulting in less performance.

To avoid this from happening, try to keep an absolute minimum of 16 kb of free memory, in addition to the 1kb already reserved by the accelerator.

In practice, this small amount of memory should not be a problem. Most cards nowadays have 2 MB of video memory, and running 1280x1024 still leaves plenty of memory unused. Even a 1600x1200 desktop will leave over 170kb unused, which will then be used by the accelerator to enhance performance.

Most 1MB cards cannot display modes larger than 1024x768 with a decent refresh rate, leaving 256kb unused.

The order in which free memory is used to accelerate certain features is as follows.

If no video memory is unused (i.e. all of it is used for display memory), no acceleration can be used at all -- not even a hardware cursor on the ET6000.

If the hardware cursor is enabled (ET6000 only) and there's at least 1kb of free video memory, 1kb is used for that.

If there is at least 1kb of free memory remaining after this, most acceleration features are enabled as well, reserving an extra 1kb of video memory.

If there's still some free memory, some extra acceleration features are enabled. These require more free video memory, depending on the virtual screen width and the color depth (bpp). The server will print out how much memory it used if it could.

If there's still some free video memory, it is used as a pixmap cache. This way, small patterns and images can be kept in the video memory so that they don't need to be transferred into the video memory each time they're needed. This is beneficial because transferring an image over the bus to the video memory takes a lot more time than letting the accelerator blit it from the pixmap cache to the display memory.

12. ET6000 memory size facts and fiction

The ET6000 uses a special kind of video memory called MDRAM (multi-bank DRAM). It may have a non-power-of-two amount of MDRAM: 2.25 or even 4.50 MB. Especially 2.25 MB MDRAM is popular, since this can support 1024x768 at 24bpp without needing 4MB of RAM.

There are a few less intuitive problems with this.

First of all, All memory above the 4 MB limit is a waste of money, because the ET6000 cannot use this memory for anything at all. There are boards with 4.5 MB around, but that extra 0.5 MB is a waste. The ET6000 can only refresh 4 MB of (M)DRAM (refresh register). It can only access 64 banks of 64KB in VGA mode (bank select register). All accelerated commands use a 22-bit address (=4MB) inside the video memory. You get the idea... There is no way for the ET6000 to use anything above the 4Mb limit.

And Secondly (more importantly): you may not have 2.25 MB at all! There have been several reports about ET6000 cards that were sold with (supposedly) 2.25 MB of MDRAM, but which turned out to be standard 2MB MDRAM cards. People have been having trouble with these all along, since sometimes the X-server used to detect this as 2.25 MB (or even 2.5 MB) due to internal chip design and also due to faulty BIOSs. This memory detection problem has been fixed now, and the server should detect the correct amount of memory.

Do NOT define the amount of memory in the XF86Config yourself, unless you are absolutely sure about the amount.

There is a simple way to determine the amount of MDRAM on your card beyond doubt.

Look at the video card. There is one large chip with 204 pins on it, which is the ET6000. One socketed rectangular chip, mostly with a sticker on it,is the BIOS. The remaining big chips are (mostly) 2 or 4 other large square chips on it with the following markings:

MDRAM MD9xy ("xy" is a two-digit number) SJ-5-100 (this may differ, but it will have the same layout)

and a nice logo next to all that with 4 diamonds and the name "MoSys" underneath.

The "xy" number tells you how much MEGABITS there are in that one chip.

The amount of RAM on the card is then:

("xy" * number_of_MDRAM_chips) / 8 Mbytes

On my board, there are two MD908 chips, which means I have

(08 * 2) / 8 = 2 MB of MDRAM.

Boards with two MD909 chips have 2.25 MB, etc.

Current MDRAM chips are MD904, MD906, MD908, MD909, MD910, MD916, MD918 and MD920.

---

13. ET6000 memory bandwidth hype and the impact on video modes

Tseng has always had wet dreams about memory bandwidth, and their press announcements about the ET6000 memory bandwidth are no exception.

They claim the ET6000 using MDRAM is capable of reaching an incredible 1.2 Gbytes/sec of bandwidth. That would surpass just about everything on the market (even SGI).

And that would be true, _if_ they actually used the fastest available MDRAMs on their boards, which they don't. The stunning 1.2 GByte mark is only reached when using 4 MDRAM chips at their max clock rate of 166 MHz. But due to design limitations, the first-generation ET6000 can only drive the memories at 92 MHz (that will change when the ET6100 and ET6300 hit the streets).

This means the max. theoretical bandwidth available on current ET6000 boards is "only" 360 MB/sec on boards with 2 MDRAM chips, and 720 MB/sec on boards with 4 MDRAM chips. And this assumes a best-case situation (=extremely long bursts -- the MDRAMs use a shared address/data bus, much like the PCI bus does). In the real world, unaligned accesses both from the PCI bus and the accelerator will reduce the effective available bandwidth. The current ET6000 boards peak out at about 225 MB/sec, with 2 or 4 MDRAMs.

Whatever you may have read in press releases, the ET6000 has a 32-bit memory bus (not 128 bits; that's only the accelerator data path within the chip, if anything). That means that, with their 16-bit busses, 2 MDRAM chips already use the full bus capacity. Having 4 memory chips on an ET6000 board will not give you extra memory bandwidth.

Memory bandwidth limits the maximum resolution you can use at a given color depth. The ET6000 RAMDAC can cope with 135 MHz in any situation. But the RAM cannot. At 32bpp (sparse 16M color mode), using a 135 MHz pixel clock would require a memory bandwidth of 135*4 = 540 MB/sec, which the current ET6000 boards simply cannot cope with. And then you still need some spare bandwidth for the PCI bus and the accelerator.

That is why some modes will be refused, depending on your MDRAM memory layout, even if the amount of memory would permit such a mode. See also the trouble shooting section to see what can happen if too little memory bandwidth is available.

14. Linear addressing and 16bpp/24bpp/32bpp modes

Currently the 16-bit (32768 or 65536 colors), 24-bit (16M colors, packed pixel), and 32-bit (16M colors, sparse) pixel support in the SVGA server requires linear addressing. This restriction may be removed in a future version, but with nearly all new cards using the PCI bus (where linear addressing poses no problem), removing the linear addressing requirement presently has a lower priority than other features. Option "linear" can be specified in a depth-specific screen section to enable linear addressing; a MemBase setting (in the device section) is probably also required on non-PCI based systems, and optionally on PCI systems that have trouble finding out for themselves where the MemBase is.

Non-PCI cards are not (or not well) supported in linear memory mode at this moment. Some of them don't support it at all, and some of the ones that do have so many address decoding bugs that it isn't feasible to provide a working solution.

For the most part, many of the accelerated features in the 8bpp server have been implemented to support 16, 24, and 32 bpp modes for the W32 and the ET6000. So although there are now up to 4 times as many bits to display, the X server shouldn't feel overly sluggish. Note also that the 24bpp and 32bpp modes are only supported on a limited set of cards, and with at least 2Mb of memory.

An ET6000 with 2.25 MB MDRAM is cheap-and-sound, since it can support exactly 1024x768 at 24bpp (using all available video memory). On all other video cards, you need at least 4MB of video memory to do this. With only 2MB of (M)DRAM, 960x720 is the best you can hope for.

In the `XF86Config` `"Screen"` section, a `"Display"` subsection must be defined for each depth that you want to run, with separate Modes and virtual screen size. Example (2Mb of video memory):

```
Section "screen"
    SubSection "Display"
        Depth 8
        Virtual 1280 1024
        ViewPort 0 0
        Modes "640x480" "800x600" "1024x768"
    EndSubSection
    SubSection "Display"
        Depth 16
        Virtual 1024 992
        ViewPort 0 0
        Modes "640x480" "800x600" "1024x768"
    EndSubSection
```


```
    SubSection "Display"
        Depth 24
        Virtual 960 720
        ViewPort 0 0
        Modes "640x480" "800x600"
    EndSubSection
    SubSection "Display"
        Depth 32
        Virtual 832 600
        ViewPort 0 0
        Modes "640x480" "800x600"
    EndSubSection
EndSection
```

Information for Tseng Chipset Users **:** *Linear addressing and 16bpp/24bpp/32bpp modes*

Previous: *ET6000 memory bandwidth hype and the impact on video modes*

Next: *Trouble shooting with the SVGA Tseng driver*

15. Trouble shooting with the SVGA Tseng driver

First of all, make sure that the default modes selected from your `XF86Config` are supported by your monitor, i.e. make sure the horizontal sync limit is correct. It is best to start with standard 640x480x256 with a 25.175 MHz clock (by specifying a single horizontal sync of 31.5) to make sure the driver works on your configuration. The default mode used will always be the first mode listed in the modes line, with the highest dot clock listed for that resolution in the timing section.

Some general hints:

- Put Option "slow_dram" in the Device Section.
- Put Option "pci_burst_off" in the Device Section.
- Put Option "w32_interleave_off" in the Device Section.
- Take out the Hercules monochrome adapter, if you have one. Many configurations of the ET4000/W32 series do not allow one in the system.
- Get a motherboard with its local bus running at 33 MHz. Many, if not all, ET4000/W32 boards will surely behave in a funny way on a 50-MHz bus. You may have to use a wait state or two, but first try without any.
- Cold-boot your machine. Do not run anything that messes with the video hardware, including other X servers, before running `XF86_SVGA`.
- In case of an ET6000 card, try specifying chipset "et6000" in the Device Section. The card normally auto-probes from the PCI bus, but on some systems, another on-board VGA card, although disabled, may cause the ET6000 server to want to use the other card.

Note that some VESA standard mode timings may give problems on some monitors (try increasing the horizontal sync pulse, i.e. the difference between the middle two horizontal timing values, or try multiples of 16 or 32 for all of the horizontal timing parameters).

There is a video signal, but the screen doesn't sync.

> You are using a mode that your monitor cannot handle. If it is a non-standard mode, maybe you need to tweak the timings a bit. If it is a standard mode and frequency that your monitor should be able to handle, try to find different timings for a similar mode and frequency combination.

Horizontal jitter at high dot clocks.

> This problem shows up especially when drawing operations such as scrolling or blitting are in progress. There is currently no easy fix for this, You can try the `"fast_dram"` option, or use a lower dot clock. If that is not sufficient, the `"noaccel"` option will almost always help (it leaves more bandwidth for the RAMDAC). In most cases, this is caused by the video memory not being able to provide pixel data to the RAMDAC fast enough, so it gets fed with garbage.

`Wavy' screen.

Horizontal waving or jittering of the whole screen, continuously (independent from drawing operations). You are probably using a dot clock that is too high; it is also possible that there is interference with a close MCLK. Try a lower dot clock (sometimes even dropping it by 0.5 MHz may work). You can also try to tweak the mode timings; try increasing the second horizontal value somewhat. Here's a 65 MHz dot clock 1024x768 mode (about 60 Hz) that might help:

```
"1024x768"     65      1024 1116 1228 1328     768  783  789  818
```

Crash or hang after start-up (probably with a black screen).

Try the `"noaccel"` option. Check that the BIOS settings are OK; in particular, disable caching of 0xa0000-0xaffff. Disabling hidden DRAM refresh may also help.

On Linux systems, if "APM" (power management) support is enabled in the kernel, the server may start up in power-save mode or with a black screen. Rebuild your kernel with APM support disabled.

Crash, hang, or trash on the screen after a graphics operation.

This may be related to a bug in one of the accelerated functions, or a problem with the BitBLT engine. Try the `"noaccel"` option. Also check the BIOS settings.

`ACL: TIMEOUT' messages from the server.

Same as for the above entry. However, on some systems, the problem will not go away no matter what you do. It may be related to the operating system you use (it has only been seen on Linux systems, and even then it depends on the kernel versions). Sometimes, choosing another MemBase may help.

Occasional erroneous pixels in text, pixel dust when moving window-frame

Probably related to MCLK setting that is too high (can happen with linear addressing even though banked mode runs OK). Most (if not all) ET6000 cards are sold with the MCLK slightly over clocked for performance (the current norm is 90 or 92 MHz), which may cause these problems. There is currently no fix for this. If the pixel dust is only temporary (it disappears as soon as nothing moves on the screen anymore), then memory bandwidth is probably the cause. The only solution is to disable acceleration, or, if that doesn't help, using a lower pixel clock.

Textmode is not properly restored

This has been reported on some configurations. Sometimes a Chipset line will fix this. Normally you should be able to restore the textmode font using a utility that sets it (`setfont`, `runx`, `restorefont` on Linux).

Mostly black or blue screen when using accelerated driver features

If you are seeing a mostly black or blue screen, with only a few icons (pixmaps) displayed, this section applies to you.

There can be several causes for this.

One is if the amount of memory is not detected (or specified) correctly. If the server's autodetection mechanism detects too much memory, accelerated features will not work. Define the

amount of memory in the `XF86Config` file. This seems to happen sometimes on some 2.25 MB ET6000 cards, where the server detects 2.5 MB instead (add `videoram "2304"` in this particular case).

If that doesn't help, disabling acceleration (option `"noaccel"`) is the only solution.

Problems with DMA hardware (floppy, tape)

On some systems, the accelerated server will interfere with other hardware that uses ISA DMA. Most notably is the PC floppy controller and sound cards. The floppy interface cannot cope with inordinately long bus-holds, which may occur during large accelerated operations. The Linux-ftape module for example (a floppy-tape driver) will generate lots of "write error" messages when running a backup or restore operation while the X-server is in use. These errors should not be fatal, but that all depends on how well the operating system handles these conditions. Linux seems to cope.

There are two possible solutions: disable acceleration using the `"noaccel"` option, or disable PCI-retry (which is causing the large bus delays) by removing the `"pci_retry"` option. This will cause a very small slowdown of accelerated operations.

The `"pci_retry"` option applies not only to the PCI bus systems, but has a similar effect on other busses.

"Cannot read colourmap from VGA. Will restore with default"

If this error occurs, the server was unable to properly initialize the RAMDAC, and tries to restore a default color map. On some unsupported RAMDACs, this will have the adverse effect of removing all color altogether, leaving you with a bunch of weird colors, or with a completely black screen. If that happens, add the `ramdac "normal"` statement to the Device section in your `XF86Config` file. In most cases, this will solve the color problem.

Why does the server report my ModeLine with only half the pixel clock?

For ET4000W32p cards at 8bpp, some modes using a clock over 75 MHz (e.g. a 1152x910 mode with 95 MHz pixel clock) will produce the following message in the Xserver output:

(--) SVGA: Mode "1152x910" will use pixel multiplexing

And later, when the accepted modelines are reported:

(**) SVGA: Mode "1152x910": mode clock = 47.500

This is normal, because with pixel multiplexing, only half the clock is needed as two pixels are sent to the RAMDAC per clock pulse.

For other screen drawing related problems, try the `"noaccel"` option.

If you are having driver-related problems that are not addressed by this document, or if you have found bugs in accelerated functions, you can try contacting the XFree86 team.

In fact, reports (success or failure) are very welcome, especially on configurations that have not been tested. You can do this via the BetaReport form (mail it to report@XFree86.org). You may want to keep an eye on forthcoming beta releases at *www.xfree86.org*.

TOC

16. Acknowledgments

Most of these stem from the old XF86_W32 server. That code was used extensively for getting the SVGA server to work on all the Tseng cards, so they are still somewhat valid.

Glenn G. Lai wrote the original XF86_W32 server. It was modified by Dirk Hohndel and Koen Gadeyne to support some more hardware.

Jerry J. Shekhel (*jerry@msi.com*) gave me (GGL) the 1-M Mirage ET4000/W32 VLB board on which the initial development (X_W32) was done.

X11R6 and The XFree86 Project provide the base code for XF86_W32.

Hercules Computer Technology Inc. lent me (GGL) a 2-M Hercules Dynamite Pro VLB board for the development that led to `XF86_W32`. They donated a Dynamite Power PCI to The XFree86 Project, that was used by DHH to extend the server.

Tseng Labs kindly donated (KMG) an ET6000-based board (a Jazz Multimedia G-Force 128), which spurred the development of the ET6000 code. They also provided an ET6100 evaluation board.

Heiko Eissfeldt provided an ET4000W32p_rev_b board which allowed us to get better support for those rev_a and rev_b boards.

Gyorgy Krajcsovits donated an ET4000W32p + CH8398 board. A Really Good Move!

Numerous testers have given me feedback for `X_W32` and later `XF86_W32`. I apologize for my failure to keep track of the people who tested `X_W32`, but the names of the people involved with the `XF86_W32` testing are listed below:

Linux:

bf11620@coewl.cen.uiuc.edu (Byron Thomas Faber)
dlj0@chern.math.lehigh.edu (David Johnson)
peterc@a3.ph.man.ac.uk (Peter Chang)
dmm0t@rincewind.mech.virginia.edu (David Meyer)
nrh@philabs.Philips.COM (Nikolaus R. Haus)
jdooley@dbp.caltech.edu (James Dooley)
thumper@hitchcock.eng.uiowa.edu (Timothy Paul Schlie)
klatta@pkdla5.syntex.com (Ken Latta)
robinson@cnj.digex.net (Andrew Robinson)
reggie@phys.washington.edu (Reginald S. Perry)
sjm@cs.tut.fi (M{kinen Sami J)
engel@yacc.central.de (C. Engelmann) **use** *cengelm@gwdg.de*
postgate@cafe.net (Richard Postgate)
are1@cec.wustl.edu (Andy Ellsworth)
bill@celtech.com (Bill Foster)

FreeBSD:

ljo@ljo-slip.DIALIN.CWRU.Edu (L Jonas Olsson)

Several people have developed code for the SVGA Tseng driver (this list is incomplete):

- Glenn G. Lai
- Dirk H. Hohndel
- Koen Gadeyne


TOC

- OEyvind Aabling
- Dejan Ilic
- Mark Vojkovich
- Harald Nordgard Hansen
- David Bateman
- Gyorgy Krajcsovits
- Kurt Olsen

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/tseng.sgml,v 3.15.2.18 1998/11/13
13:00:48 dawes Exp $
```

```
$XConsortium: tseng.sgml /main/6 1996/10/27 11:06:09 kaleb $
```

Information for Tseng Chipset Users **:** *Acknowledgments*
Previous: *Trouble shooting with the SVGA Tseng driver*
Next: *Information for Tseng Chipset Users*

Notes on the AGX Server

Henry Worth

24 June 1995

1. General Notes
2. Acknowledgments
3. Known Problems
4. ToDo
5. XF86Config
6. Xga configuration

1. General Notes

This server currently supports the IIT AGX-016, AGX-015, AGX-014 and XGA-2 chipsets. The AGX chipset is based on XGA architecture, but is missing several features and differs on others. There's also untested support for the XGA-1 and AGX-010 chipsets. Pixel depths of 8bpp, 15bpp, 16bpp are generally supported. Unpacked 24bpp (RGBX 32bpp) is not yet stable enough to release.

RAMDACs currently supported are the Brooktree (BT481, BT482, and BT485) and AT&T (20C505) RAMDACs used by the Hercules Graphite series, and Sierra RAMDACs (15025 and 15021), and Generic VGA RAMDAC. Untested support has been added for the AT&T 20C490 series.

The current driver has a number of acceleration routines: solid and dashed zero-width lines (except AGX-014), bitblt fills, tiles, and stipples, solid arc and polygon fills, character glyphs and font cache for 8-bit characters.

Boards that have had some testing include ISA and VLB versions of most of the Hercules Graphite series, Spider Black Widow VLB and Black Widow Plus VLB, Boca Vortek VL, CatsEye/X XGA-2, and the PS/2-57 planar XGA-2. The Orchid Celsius is very similar to the Spider and Boca boards, except some batches may use one of the AT&T 20C490 series RAMDACs, instead of the Sierra 15025. There has also been a report of a generic board that uses a UMC RAMDAC that may be an AT&T 20C490 Clone.

2. Acknowledgments

First, to Hercules Customer Support for providing a loaner board to get things started.

Second, to the XFree86 team, and those who who have contributed to their efforts to the project, for the foundation of work that provided a basis for bootstrapping this server.

3. Known Problems

- The accelerated line routines don't match lines written by the mi/cfb routines. This is noticeable when switching between virtual consoles while running routines that draw and erase lines. Seems to have been reduced/fixed in previous releases but need more testing.
- Some special-case speedup added to cached font rendering in 3.1.1 has been disabled as is over-aggressive in some cases. This cuts the performance on terminal-fonts in half, and font performance is already low for the AGX chips compared to their contemporaries.
- As in all software, needs more testing.

4. ToDo

- Address the above known problems.
- Additional acceleration routines and general performance improvements. Many existing acceleration routines are Q&D adaptations of existing routines from other servers that support graphics chips that differ significantly, architecturally, from that XGA and are undoubtedly less than optimal. In particular some of the general per-operation overhead to set-up the graphics context should be moved to the ValidateGC() routines.
- Complete HW cursor support, most of the code is done (or borrowed from other servers). There just remains a little setup code and then finding a lot of time to debug and test the numerous permutations.
- Complete support for the Graphite Pro's 84-pin RAMDAC. (the 2MB version of the Graphite Pro has both RAMDACs, the 1Mb only the 44-pin RAMDAC). Currently, the 84-pin RAMDAC is only supported in clock-doubled pixmux mode, the server will switch between RAMDACs as required by the video mode In >8bpp modes this switching does not occur.
- Implement more HW probing, this will be difficult as it appears some (all?) AGX-based vendors don't implement the VESA VXE POS registers, although the AGX chip does support it (and some vendors claim VXE compliance...). There are a few rev/vendor registers in the AGX chip but they are not documented. Note: SuperProbe also does not support probing for AGX/XGA chips. ISA POS probing is supported for the XGA chips and some code for EISA POS is also included but not tested.
- Micro-optimizations, in particularly reducing processing overhead for common special cases that don't require full generality.

5. XF86Config

Device Section Entries and Options Currently Supported:

The minimum that must be specified in the XF86Config device section for the AGX-014, AGX-015, AGX-016, and ISA-based XGA-1 and XGA-2 is the Chipset. However to get full capability out of the AGX-01[456] chips, the RAMDAC should be specified. Other parms may select additional capabilities, or may used to override the defaults or reduce start-up time be suppressing probing. XGA specific configuration is covered at the end of this document. The XGA entries can generally be used to override defaults for the AGX-01[456] as well.

Ramdac

Be sure to check the clock rating of the RAMDAC(s) on your video board and don't exceed that rating even if the server allows it, overclocking RAMDACs will damage them.

The clock rating generally appears as a suffix to the part number, may only have the most significant digit(s), and may be mixed with other codes (e.g. package type). For example, an 85MHz Bt481 in a plastic J-lead package has a part number of Bt481KPJ85 and a 135MHz AT&T20C505 has a part number of ATT20C505-13. Sierra stamps the rated speed below the part numbers in a dark ink.

"normal"

normal VGA style RAMDAC (6-bit DAC), default if none specified. Most boards should work with this parm, but some capabilities will be unavailable. Only 8bpp is available.

"bt481"

bt481 RAMDAC (supports 8-bit DAC)

"bt482"

bt482 RAMDAC (supports 8-bit DAC) The Hercules Graphite HG210 uses the BT481 or BT482, the only difference between these two is the BT482's HW cursor (not yet supported). The BT481/2 are limited to 85Mhz. 8bpp, 15bpp, 16bpp are supported.

"ATT20c490"

AT&T490 RAMDAC (includes 49[123] - supports 8-bit DAC). Limited to 110Mhz at 8bpp. 8bpp, 15bpp, and 16bpp are supported.

"SC15025"

Sierra SC15025 and SC15021 RAMDAC (support 8-bit DAC). The SC15025 is limited to 125Mhz, and the SC15021 135Mhz. Check the RAMDAC's actual rating, some SC15025's used in AGX based boards are only rated to 110Mhz. 8bpp, 15bpp, and 16bpp are supported.

"herc_dual_dac"

Hercules Graphite Pro RAMDAC probe. If the 84-pin Big-RAMDAC is installed (2MB models), will use the Big RAMDAC, but only clocks-doubled, pixel- multiplexed modes (higher clock values only!). Lower clocks and resolutions in 8bpp mode are supported by switching to the Small 44-pin RAMDAC. 15bpp and 16bpp are supported.

There has been one report of the "dac-8-bit" option not working with a Graphite Pro equipped with a BT485 RAMDAC, puzzling since it should be identical to the AT&T20C505 in this regard. No startup messages or XF86Config were submitted to aid problem isolation.

Not supported by the HG210 Graphite.

"herc_small_dac"

Hercules Graphite Pro RAMDAC probe. Forces use of only the BT481/482 RAMDAC. 8bpp, 15bpp, 16bpp, and unpacked 24/32bpp are supported.

Not supported by the HG210 Graphite.

"xga"

To allow overriding the default VGA style RAMDAC control for the AGX-010.

Ramdac related Option Flags:

"dac_6_bit"

Sets RAMDAC to VGA default 6-bit DAC mode (default for "normal").

"dac_8_bit"

Sets supported RAMDAC's to 8-bit DAC mode (default for all but "normal").

"sync_on_green"

Composite sync on green for RAMDAC's that support this feature (BT481/481 and AT&T20c490). However, whether any boards have necessary traces and glue logic is doubtful.

Chipset:

Must be specified, possible values: "AGX-016", "AGX-015", "AGX-014", "AGX-010", "XGA-2", or "XGA-1". Some AGX vendors place stickers over the chip, in general, if it's a VLB board it's probably an AGX-015 and if it's an ISA board it may be an AGX-014. The Hercules Graphite Power Pro and Spider Black Widow Plus use the AGX-016 chipset. In general, specifying a lower revision in the AGX-0{14,15,16} series does not seem to causes problems (except lower performance from the AGX-014's non-accelerated line drawing).

Note: Only the AGX-016, AGX-015, AGX-014 and XGA-2 have had any testing. Most of the development has been with an AGX-015 based 2MB Hercules Graphite VL PRO (HG720) and most of testers for previous releases had AGX-014 based 1MB Hercules Graphite (HG210).

The limited documentation I have for the AGX-010 is that is is a clone of the XGA architecture with a few additional configuration registers. What is not clear is whether to use XGA or extended-VGA RAMDAC control registers. The post-3.1.1 default is now VGA control registers, but XGA control registers can be forced with the XGA RAMDAC parm. Likewise the configuration parms described in the XGA section can be used to override the AGX defaults for I/O and memory addresses.

VideoRam:

Will be probed if not specified. The startup will be a little faster if specified.

Tuning Option flags:

Bus I/O interface:

"8_bit_bus"

Force 8-bit I/O bus.

"wait_state", "no_wait_state"

Set or clear CPU access wait state, default is the POST setting.

"fifo_conserv"

Disable Memory I/O Buffer, AGX-015 and AGX-016. MS-Windows driver default. Required by some VLB systems with `aggressive timing'. The default for this server is to disable the buffer.

"fifo_moderate"

Enable the AGX-015/016's Memory I/O buffer.

"fifo_aggressive"

Enable the AGX-016's extra-large buffer. Either option may result in garbage being left about the s disabled by default. A good test is the xbench or x11perf dashed lines tests, if random dots are dra fifo_conserv is required. So far, no boards have been reported that worked correctly with the buffers enabled.

Memory Timing:

POST defaults should be ok.

"vram_delay_latch", "vram delay_ras", "vram_extend_ras"

Vram timing options.

"slow_vram", "slow_dram"

Set all of the vram timing options.

"med_dram"

Set vram latch delay, clear others.

"fast_vram", "fast_dram"

All of the vram timing options are cleared. Should be specified if directly specifying VRAM options in order to clear POST settings.

Debugging:

These shouldn't generally be required:

"noaccel"

(AGX,XGA) Disable Font Cache.

"crtc_delay"

(AGX) Force XGA mode CRTC delay.

"engine_delay"

AGX-015 only? adds additional VLB wait state.

"vram_128", "vram_256"

Sets VRAM shift frequency, vram_128 is for 128Kx8 VRAM. Default is to leave this bit unchanged from POST setting.

"refresh_20", "refresh_25"

Number of clock cycles between screen refreshes. Default is to leave this bit unchanged from POST setting.

"screen_refresh"

Disable screen refresh during non-blanked intervals, AGX-016. Default is leave them enabled.

"vlb_a", "vlb_b"

VLB transaction type, default is to leave this bit unchanged from POST value.

Virtual resolution:

The server now accepts any virtual width, however the actual usable CRTC line width is restricted when using the graphics engine and depends upon the chip revision. The CRTC line width and not the virtual width determine the amount of memory used. The server currently does not make use of any of the unused CRTC line's memory. CRTC line width is restricted by the following rules:

AGX-014 : 512, 1024 and 2048. (also AGX-010)

AGX-015 : 512, 1024, 1280, and 2048.

AGX-016 : 512, 640, 800, 1024, 1280, and 2048.

XGA,AGX-010 : 512, 640, 800, 1024, 1280, 1152, and 2048.

When panning I occasionally get streaks if the virtual resolution is much greater than the physical resolution. Moving the mouse a little makes it disappear. The Hercules manual indicates this also happens with the MS-Windows drivers.

The server requires at least a 64KB scratchpad (16KB for XGA's). Additional memory is useful for font cache and a larger scratchpad.

AGX Clocks:

Probing is supported, but of course the usual warnings and disclaimers apply. Probing may momentarily subject your monitor to sweep frequencies in excess of its rating. The cautious may wish to turn off the monitor while the probe is running.

Once clocks are known, they can be entered into XF86Config, then subsequent runs won't probe clocks and will be quicker to startup. For the clock probe it is recommended that the X server be run with the -probeonly option. The values in the clocks statement are the hardware input clocks and correspond to the pixel clock only at 8bpp in direct-clocking RAMDAC modes. The server will divide/multiply those values as appropriate for the RAMDAC modes available at the current pixel depth. The available pixel clocks will be displayed in the startup messages.

For the 2MB Hercules Graphites, with the "herc-dual-dac" RAMDAC specified, earlier versions of the server generated an additional 16 clocks with values doubled and some zeroed. Those are no longer needed and you should re-probe and re-enter the clock values to ensure all clocks are available to you.

The AGX-015 2MB Hercules Graphite VL Pro with an ICS1494M 9251-516 clock chip has probed clock values of:

```
25.18  28.80  32.70  36.00  40.00  45.00  50.40  64.70
70.10  76.10  80.60  86.30  90.40  95.90 100.70 109.40
```

Actual values according to Hercules are:

```
25.175 28.322 32.512 36.000 40.00 44.90 50.35 65.00
70.00  75.00  80.00  85.00  90.00 95.00 100.0 108.0
```

These are the values to be used in the clock statement if specifying the "normal", "bt481", or "herc_small_dac" RAMDAC in your XF86Config and your clockchip matches that above.

Clock probing assumes that the first clock is 25.175Mhz and uses that to derive the rest. A warning is displayed if the second is not near 28.322Mhz. If this warning appears, you should not use the probed clock values without additional verification from other sources.

In the case of the AGX-014 and later AGX's, only the external clock select lines are used, this means the clock values correspond to the values of the video board's clock chip.

For the AGX-010, the first 8 clocks use the standard XGA internal clock selects and the second 8 are based on AGX extensions. For the XGA-1 only 8 clocks are available. The XGA-2 uses a programmable clock and no clocks or clockchip line is required.

The maximum pixel clock generally allowed is 85MHz, but some RAMDACs support higher values. In any case you, should check your RAMDAC, some RAMDACs used on AGX based boards are produced in versions rated to lesser values than the server assumes. You should check the rating and limit yourself to that value.

Modes:

One difference I've noted from the Mach8, is that the AGX's CRTC doesn't like the start of the horizontal sync to be equal to horiz blank start (vert sync may have the same problem, I need to test some more). Interlaced and +/-sync flags are supported but have had very little testing. For interlaced modes make sure the number of lines is an odd number.

The doublescan flag is now supported, however the minimum clock supported is generally 25MHz, so resolutions of less than 400x300 are not likely to be supported by most monitors. In creating doublescan mode timings, the vertical

timings will match the apparent resolutions, e.g. for 400x300 the timings should describe 300 lines, not 600.

Examples:

For the Hercules HG720 (2MB VLB AGX-015, with BT481 and AT&T20C5050 RAMDACs), I use the following XF86Config "Device" section:

```
Section "Device"
    Identifier "HG720"
    VendorName "Hercules"
    BoardName  "Graphite VL Pro"
    Chipset    "AGX-015"
    Clocks     25.2  28.3  32.5  36.0  40.0  45.0  50.4  65.0
               70.00  75.00  80.00  85.00  90.00 95.00 100.0 108.0
    Videoram   2048
    RamDac     "herc_dual_dac"
    Option     "dac_8_bit"
    Option     "no_wait_state"
EndSection
```

For the Spider Black Widow Plus (2MB VLB AGX-016, with Sierra SC15021 RAMDAC):

```
Section "Device"
    Identifier "SBWP"
    VendorName "Spider"
    BoardName  "Black Widow Plus"
    Chipset    "AGX-016"
    Clocks      25.2  28.3  39.9  72.2  50.0  76.9  36.1  44.8
                89.0  119.8 79.9  31.5 110.0  64.9  74.9  94.9
    Videoram   2048
    RamDac     "SC15025"
    Option     "dac_8_bit"
    Option     "no_wait_state"
EndSection
```

6. Xga configuration

This server now has tested support for XGA-2 compatible boards (aka. XGA-NI). The main issue for XGA-1 support is whether clock probing works. At this time probing for board configuration is limited and detailed configuration may need to be done manually.

By default the ISA POS register will be performed. If the XGA Instance number is specified the scope of probing will be narrowed a bit. To override or disable probing, a minimum of the Instance, COPbase, and MEMbase must be specified in the XF86Config device section for the XGA card. MCA probing is not supported.

Instance nn

XGA instance number (0-7).

IObase nnnn

The I/O address of the the XGA general control registers. The standard, and default, is 0x21i0, where i is the instance number.

MEMbase nnnn

The XGA display memory address (the address the XGA coprocessor uses for video memory). This is also the system memory address of the linear aperture on boards that support it.

POS register 4 bits 7-1 contains bits 31-25 of the XGA's display memory address. Bits 24-22 of of the display memory address contains the XGA instance number. Bit 0 of POS register 4 is not used by this server as the XGA's linear aperture is not used. However, the coprocessor must still be configured with this.

The AGX-01[456] chips have a fixed display memory address.

COPbase nnnnnn

Address of the graphics engine's memory mapped control registers.

Typically:

0xC1C00 + (ext_mem_addr * 0x2000) + (instance * 0x80)

where ext_mem_addr is the high order 4-bits of POS register 2 (0-16 the server assumes zero).

The AGX-01[456] chips support 0xB1F00 (default) and 0xD1F00.

BIOSbase nnnnnn

Address of the XGA BIOS (not VGA BIOS). Can be specified as an alternate to COPbase.

Typically:

0xC0000 + (ext_mem_addr * 0x2000)

where ext_mem_addr is the high order 4-bits of POS register 2 (0-16 -- the server assumes zero).

VGAbase nnnn

Can be used to override the default 0xA0000 address for the 64KB video memory address used by the server. The only values acceptable are 0xA0000 and 0xB0000. VGA text mode restore does not work under Linux if 0xB0000 is specified.

AGX-01[456] also default to 0xA0000.

POSbase nnnn

Can be used to specify an alternate POS register probe address base from the ISA default of 0x100. The VESA VXE standard for EISA is 0xzC80, where z is the slot number).

A value of zero will disable POS register probing (required for MCA).

DACspeed nnnn

Can be used to override the servers default maximum Pixel Clock for XGA-2 of 80Mhz. The limit can be raised as high as 90Mhz, or set to lower values.

An alternate way to determine the POS register values is with the setup/diag programs that should have been included with your video board, or possibly from jumper values.

The XGA-2 has programmable clocks up to 90MHz, however at 1024x768, 72MHz is generally the max that will produce a stable display with the CatsEye/XGA-2 used for testing (IBM coprocessor and INMOS RAMDAC/serializer). Higher clocks will often generate artifacts at the top and left edges of the screen. Such artifacts can sometimes be tuned out by increasing the vertical and horizontal blanking intervals or slightly changing the clock. At pixel clock rates above 80Mhz I have seen the chip lose sync after running for several minutes, so 80Mhz has been set as the default limit for XGA-2 pixel clocks. I don't have specs on actual limits, and as there are a number of different XGA chipsets, you should use the modes documented in your owner's manual as a guide to max refresh rates. No clocks or clockchip parm are required to specify use of programmable clocks for the XGA-2.

8bpp and 16bpp are supported for the XGA-2.

For XGA-1 cards the clocks must be specified as for the AGX chips, it is not known whether the clockprobing will work. Some XGA-1 chips may support 16bpp.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/agx.sgml,v 3.19.2.1 1998/11/07
13:37:50 dawes Exp $
```

```
$XConsortium: agx.sgml /main/9 1996/10/19 18:03:50 kaleb $
```

Information for ARK Logic Chipset User

Harm Hanemaayer (*H.Hanemaayer@inter.nl.net*)

17 January 1997

1. Supported chipsets

2. Supported RAMDACs

3. Acceleration

4. Basic configuration

5. Features that may be expected in upcoming beta releases

6. Tested configurations.

1. Supported chipsets

The "ark" driver in the SVGA server is for ARK Logic graphics chipsets. The following chipsets are supported:

ARK1000PV (ark1000pv)

Chipset with 32-bit DRAM interface, supports fast DRAM timing, for VESA and PCI bus. Has powerful "coprocessor" for graphics acceleration. The max supported resolution/refresh depends on the RAMDAC used on the card; expect 256 colors up to 80 or 110 MHz dot clock; 16bpp is also supported, as is 24bpp (packed).

ARK1000VL (ark1000vl)

Older chip, VLB only. More or less compatible with ARK1000PV. It has is not been tested. You may have to disable acceleration and linear addressing.

ARK2000PV (ark2000pv)

64-bit version of the ARK1000PV. Note that an ARK2000PV equipped with 1Mb of DRAM is about equivalent to the same card with an ARK1000PV chip; 2Mb is required for 64-bit operation. Again the RAMDAC used on the card determines the max supported dot clocks. At 8bpp, multiplexing over a 16-bit RAMDAC path is not yet supported so expect dot clocks up to 110 MHz; 16bpp and 32bpp are supported, as well as experimental packed 24bpp, depending on the RAMDAC.

ARK2000MT (ark2000mt)

This is a newer chip, compatible with the AR2000PV.

The ARK2000MI is not yet supported.

The chipset may not be detected automatically. In this case use a line like `Chipset "ark1000pv"` in the `Device` section of the XF86Config file. Any options must also be specified in this section.

2. Supported RAMDACs

If no RAMDAC is specified, a standard RAMDAC supporting 256 colors up to 80 MHz dot clock frequency is assumed. The following RAMDAC types can be specified in the Device section of the XF86Config file (e.g. `Ramdac "att20c490"`):

att20c490

Industry-standard 8-bit RAMDAC. The RAMDAC used on the basic Hercules Stingray Pro is compatible. 16bpp color depth is supported up to 40 or 55 MHz, depending on the DAC speed rating. Packed 24bpp is supported up to about 36 MHz.

att20c498

Industry-standard 16-bit RAMDAC. The RAMDAC used on the Hercules Stingray Pro/V and the Stingray 64/V is compatible. 16bpp is supported up to 80 MHz or 110 MHz dot clock frequency, 32bpp is supported up to 40 or 55 MHz.

zoomdac

This is the actual DAC used by the Hercules Stingray Pro/V and 64/V. It is treated mostly as an ATT20C498, but with dot clock limits set correctly (16bpp up to 55 MHz with ARK1000PV, up to at least 110 MHz with ARK2000PV). In addition, packed 24bpp is supported (up to 36 MHz with ARK1000PV, not yet on the ARK2000PV), and 32bpp is also supported on the ARK2000PV (up to 55 MHz) This RAMDAC should be auto-detected.

stg1700

Completely untested.

ics5342

This is a clockchip/RAMDAC combination and is used on the Diamond Stealth 64 Graphics 2001 and newer Hercules cards that use the ARK2000MT. It is supported at 16bpp and 32bpp in addition to 256 color mode. 32bpp mode may not work.

The Dacspeed keyword can be used to indicate the speed rating of the RAMDAC, but it must be used with care. The raw clock frequency may exceed 80 MHz. Both the ARK chips and some of the RAMDACs are specified for raw speeds up to 120 MHz, but this might violate FCC regulations or otherwise be unstable. High dot clock 8bpp modes (e.g. 135 MHz) are normally achieved by sending 2 pixels at at time over a 16-bit DAC path (the raw clock would be 67.5 MHz for 135 MHz dot clock), a mode of operation that is not yet supported by this driver. No high-dot clock configurations have been tested.

The driver now limits the maximum dot clocks according to the DRAM speed (bandwidth). Because it is not possible to determine the memory clock speed (except on the ICS5342), the driver assumes a default of 60 MHz. On an ARK1000PV, that allows 8bpp up to 109 MHz, 16bpp up to 55 MHz, 24bpp up to 36

MHz, and 32bpp up to 27 MHz. On an ARK2000PV with 2MB memory, it allows 16bpp up to 110 MHz, 24bpp up to 72 MHz, and 32bpp up to 55 MHz. If you know what your real memory clock is, you can specify it with the `MCLK` keyword, for example `MCLK 70`.

To run `XF86_SVGA` at 16 bpp, pass options to the X server as follows:

```
startx -- -bpp 16                5-6-5 RGB ('64K color', XGA)
startx -- -bpp 16 -weight 555    5-5-5 RGB ('Hicolor')
startx -- -bpp 24                8-8-8 RGB (packed 24-bit truecolor)
startx -- -bpp 32                8-8-8 RGB (32-bit pixel truecolor)
```

Information for ARK Logic Chipset Users **:** *Supported RAMDACs*
Previous: *Supported chipsets*
Next: *Acceleration*

3. Acceleration

The driver takes full advantage of the new XAA (XFree86 Acceleration Architecture) in the SVGA server. In fact the ARK driver was the initial XAA development platform. Most common graphics operations are accelerated, including most types of rectangular and non-rectangular filling, screen-to-screen BitBLTs, line drawing, and text and bitmap expansion. Expect over 300k xstones on a 2MB ARK2000PV/MT.

At 24bpp, acceleration is less complete, but curiously, greyscale colors permit faster drawing. If you suspect a problem with acceleration, use the "noaccel" option. If text or bitmaps do not seem to be rendered correctly, you could try the "xaa_no_col_exp" option. To disable the pixmap cache, use "no_pixmap_cache".

The hardware cursor is disabled by default. With unmodified mode timings, there used to be two horizontal lines and a band following the mouse pointer over the screen. The driver now automatically modifies the mode timing to eliminate this effect; this has not been tested on all possible configurations. Use the "hw_cursor" option to enable the hardware cursor.

Linear addressing is the default mode of operation. If the server does not start correctly, you may want to try the "no_linear" option.

The older ARK1000VL is probably not compatible with acceleration. Use the "noaccel" and "no_linear" options.

4. Basic configuration

It is recommended that you generate an XF86Config file using the `XF86Setup' or `xf86config' programs, which should produce a working high-resolution 8bpp configuration, although the modelines might need reshuffling for optimal screen refresh. You may want to include mode timings in the `Monitor` section that better fit your monitor (e.g. 1152x864 modes).

In order to prevent stress on your monitor, it is recommended that you turn off your monitor during clock probing (X -probeonly), which also happens if you start the server with no Clocks line present in the Device section of the XF86Config. The following Clocks line can be used for the Hercules Stingray Pro, Pro/V and older 64/V using an ARK Logic clock generator (so there's no need to probe clocks for this card, just insert the following line in the Device section of the XF86Config file):

```
Clocks 25.175 28.3 40 72 50 77 36 44.9
Clocks 128.43 118.8 80 31.5 110 63.96 74.19 95
```

The higher frequencies have not been tested, there might be a mismatch in the 60-80 MHz range.

TOC

5. Features that may be expected in upcoming beta releases

- Support for high dot clocks (>80 MHz, up to 135 MHz) at 8bpp by sending two pixels at a time over a 16-bit RAMDAC path on an ARK2000PV/MT with supported RAMDAC.
- Support for packed-24bpp mode up to 72 MHz on an ARK2000PV with ZoomDAC.
- The acceleration may be further optimized and stabilized.
- Existing problems may be fixed.
- Support for the ARK2000MI, if it materializes.

6. Tested configurations.

Hercules Stringray Pro (ARK1000PV + ATT20C490-compatible RAMDAC)

Supported at 8bpp, 16bpp and 24bpp. Fixed set of clocks. There seems to be a restriction to the mode timings at 24bpp; the last horizontal number (HTotal) must be divisible by 4 but not by 8. If the modeline is wrong, the colors would be incorrect. The driver automatically corrects the mode timing.

Hercules Stingray Pro/V (ARK1000PV + IC Works ZoomDAC)

Supported at 8bpp, 16bpp and 24bpp. Fixed set of clocks. The same restrictions above exist for the 24bpp mode. Problems with textmode restoration have been reported on some OS's.

Hercules Stingray 64/V (ARK2000PV + IC Works ZoomDAC)

Supported at 8bpp, 16bpp and 32bpp. Fixed set of clocks. Problems with textmode restoration have been reported on some OS's.

Hercules Stingray 64 with ARK2000MT + ICS5342 Clockchip/RAMDAC

This may also apply to other cards with the ICS5342, such as the Diamond Stealth 64 Graphics 2001. Use RAMDAC "ics5342". Programmable clockchip (don't specify any Clocks lines). Supported at 8bpp, 16bpp and 32bpp. 32bpp has been reported not to work. This configuration has not been tested with a post-3.2 server.

If are having driver-related problems that are not addressed by this document, you can try contacting the XFree86 team (the current driver maintainer can be reached at H.Hanemaayer@inter.nl.net), or post in the Usenet newsgroup comp.windows.x.i386unix.

In fact, reports (success or failure) are very welcome, especially on configurations that have not been tested. You can do this via the bug report form (or send mail to XFree86@XFree86.org). You may want to keep an eye on forthcoming beta releases at the XFree86 web site.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/ark.sgml,v 3.9 1997/01/25 03:22:20
dawes Exp $
```

```
$XConsortium: ark.sgml /main/6 1996/10/28 05:24:04 kaleb $
```

ATI Adapters README file

Marc Aurele La France

1999 June 23

This is the README for the XFree86 ATI driver included in this release.

---

1. Statement of intent

2. A note on acceleration

3. Current implementation for ATI adapters

4. Current implementation of generic VGA support for non-ATI adapters

5. XF86Config specifications

    5.1. ChipSet "name"
    5.2. Clocks
    5.3. Option "nolinear"
    5.4. MemBase address
    5.5. Modelines

6. Known problems and limitations

7. Reporting problems

8. Driver history

9. Driver versions

1. Statement of intent

Generally speaking, the driver is intended for all ATI video adapters, providing maximum video function within hardware limitations. The driver is also intended to optionally provide the same level of support for generic VGA or 8514/A adapters. This driver is still being actively developed, meaning that it currently does not yet fully meet these goals.

The driver will provide

- accelerated support if an ATI accelerator is detected *and* the user has not requested that this support be disabled; otherwise
- accelerated support if a non-ATI 8514/A-capable adapter is detected *and* the user has requested such support; otherwise
- unaccelerated SuperVGA support if an ATI VGA-capable adapter is detected; otherwise
- generic VGA support if a non-ATI VGA-capable adapter is detected *and* the user has requested such support.

Thus, the support provided not only depends on what the driver detects in the system, but also, on what the user specifies in the XF86Config file. See the "XF86Config specifications" section below for details.

If none of the above conditions are met, the ATI driver will essentially disable itself to allow other drivers to examine the system.

2. A note on acceleration

The meaning of "acceleration", as used in this document, needs to be clarified. Two of the many components in an accelerator are the CRT controller (CRTC) and the Draw Engine. This is in addition to another CRTC that, generally, is also present in the system (often in the same chip) and typically provides EGA, VGA or SuperVGA functionality.

A CRTC is the component of a graphics controller that is responsible for reading video memory for output to the screen. A Draw Engine is an accelerator component that can be programmed to manipulate video memory contents, thus freeing the CPU for other tasks.

When the VGA CRTC is used, all drawing operations into video memory are the responsibility of the system's CPU, i.e. no Draw Engine can be used. On the other hand, if the accelerator's CRTC is chosen to drive the screen, the Draw Engine can also be used for drawing operations, although the CPU can still be used for this purpose if it can access the accelerator's video memory.

Video acceleration refers to the programming of an accelerator's Draw Engine to offload drawing operations from the CPU, and thus also implies the use of the accelerator's CRTC.

3. Current implementation for ATI adapters

The driver currently supports the SuperVGA capabilities of all ATI adapters except some early Mach8 and Mach32 adapters that do not provide the required functionality. This support works for monochrome, 16-colour and 256-colour video modes, if one of the following ATI graphics controller chips is present:

```
VGAWonder series:  18800, 18800-1, 28800-2, 28800-4, 28800-5, 28800-6
   Mach32 series:  68800-3, 68800-6, 68800AX, 68800LX
   Mach64 series:  88800GX-C, 88800GX-D, 88800GX-E, 88800GX-F, 88800CX,
                   264CT, 264ET, 264VT, 264GT (3D Rage), 264VT-B, 264VT3,
                   264VT4, 264GT-B (3D Rage II), 3D Rage IIc, 3D Rage Pro,
                   3D Rage LT, 3D Rage LT Pro, 3D Rage XL, 3D Rage XC
```

The driver also supports 32K, 64K and 16M-colour modes on the 264xT and 3D Rage series of adapters using the accelerator CRTC (but not the VGA CRTC). This support is as yet unaccelerated.

The newer Rage 128 chips are not yet supported.

Adapters based on the above chips have been marketed under a rather large number of names over the years. Among them are:

```
VGAWonder series:  VGAWonder V3, VGAWonder V4, VGAWonder V5, VGAWonder+,
                   VGAWonder XL, VGAWonder XL24, VGAWonder VLB, VGA Basic,
                   VGA Basic 16, VGA Edge, VGA Edge 16, VGA Integra,
                   VGA Charger, VGAStereo F/X, VGA 640, VGA 800, VGA 1024,
                   VGA 1024D, VGA 1024 XL, VGA 1024 DXL, VGA 1024 VLB
    Mach8 series:  Graphics Ultra, Graphics Vantage, VGAWonder GT
                   (None of the 8514/Ultra and 8514 Vantage series is
                    supported at this time)
   Mach32 series:  Graphics Ultra+, Graphics Ultra Pro, Graphics Wonder,
                   Graphics Ultra XLR, Graphics Ultra AXO, VLB mach32-D,
                   PCI mach32-D, ISA mach32
   Mach64 series:  Graphics Xpression, Graphics Pro Turbo, WinBoost,
                   WinTurbo, Graphics Pro Turbo 1600, Video Xpression,
                   3D Xpression, Video Xpression+, 3D Xpression+,
                   3D Charger, Video Charger, WinCharger, All-In-Wonder,
                   All-In-Wonder PRO, 3D Pro Turbo, XPERT@Play,
                   XPERT@Play 98, XPERT@Work, XPERT 98, XPERT LCD,
                   XPERT XL
```

VGAWonder, Mach8 and Mach32 ISA adapters are available with or without a mouse.

These adapters are available with a variety of clock generators and RAMDACs. The 264xT and 3D Rage series of chips are integrated controllers, meaning that they include a programmable clock generator and a

RAMDAC. See the "XF86Config specifications" section below for details.

This driver still does not provide support for accelerated drawing to the screen. This means that all drawing is done by the CPU, rather than by any accelerator present in the system. This can make opaque moves, for example, quite "jerky". Thus, given that IBM 8514/A and ATI Mach8 do not allow CPU access to their frame buffer, the driver will currently ignore these accelerators. Most Mach32 adapters provide both accelerated function and VGA functionality, but the driver currently only uses the VGA.

The driver *does* however support the accelerator CRTC present in all ATI Mach64 adapters. For 256-colour, and higher depth modes, this support will be used by default, although an XF86Config option can be specified to use the SuperVGA CRTC instead. A linear video memory aperture is also available in 256-colour and higher depth modes and enabled by default if a 264xT or 3D Rage controller is detected or, on 88800 controllers, if the accelerator CRTC is used. XF86Config options are available to disable this aperture, or (on non-PCI adapters) enable it or move it to some other address.

ATI Adapters README file **:** *Current implementation for ATI adapters*
Previous: *A note on acceleration*
Next: *Current implementation of generic VGA support for non-ATI adapters*

4. Current implementation of generic VGA support for non-ATI adapters

Support for generic VGA with non-ATI adapters is also implemented, but has undergone only limited testing. The driver will intentionally disallow the use of this support with ATI adapters. This support must be explicitly requested through an XF86Config ChipSet specification. This prevents the current generic driver from being disabled.

This driver's generic VGA support is intended as an extension of that provided by the current generic driver. Specifically, within the architectural bounds defined by IBM's VGA standard, this driver will allow the use of any 256-colour mode, and any dot clock frequencies both of which allow for many more mode possibilities.

The driver will enforce the following limitations derived from IBM's original VGA implementation:

- There can only be a set of four (non-programmable) clocks to choose from.
- Video memory is limited to 256kB in monochrome and 16-colour modes.
- Video memory is limited to 64kB in 256-colour modes.
- Interlaced modes are not available.

5. XF86Config specifications

Except for clocks, the driver does not require any XF86Config specifications of its own for default operation. The driver's behaviour can however be modified by the following specifications.

5.1. ChipSet "name"

The default ChipSet name for this driver is "ati".

If "ativga" is specified instead, the driver will not use any ATI accelerator CRTC it detects, relying instead on any detected ATI VGA CRTC to provide the screen image.

A ChipSet name of "ibmvga" enables the driver's generic VGA support, but only for non-ATI adapters. If an ATI adapter is detected, the driver will operate as if "ativga" had been specified instead.

For compatibility with other XFree86 servers, both past and present, that support ATI adapters, the driver also recognizes "vgawonder", "mach8", "mach32" and "mach64" as chipset names. In this version of the driver, all such names are equivalent to "ati". In some future release, each name will have a different meaning to be documented at that time.

5.2. Clocks

For the purpose of specifying a clock line in your XF86Config, one of four different situations can occur, as follows.

Those configuring the driver's generic VGA support for a non-ATI adapter, can skip ahead to the "Clocks for non-ATI adapters" section below. Those *not* trying to configure the driver for a Mach64 adapter, can skip ahead to the "Clocks for fixed clock generators on ATI adapters" section below.

The very earliest Mach64 adapters use fixed (i.e. non-programmable) clock generators. Very few of these (mostly prototypes) are known to exist, but if you have one of these, you can also skip ahead to the "Clocks for fixed clock generators on ATI adapters" section below.

The two cases that are left deal with programmable clock generators, which are used on the great majority of Mach64 adapters.

If you are uncertain which situation applies to your adapter, you can run a clock probe with the command "X -probeonly".

5.2.1. Clocks for supported programmable clock generators

At bootup, video BIOS initialization programmes an initial set of frequencies. Two of these are reserved to allow the setting of modes that do not use a frequency from this initial set. One of these reserved slots is used by the BIOS mode set routine, the other by the particular driver used (e.g. MS-Windows, AutoCAD,

X, etc.). The clock numbers reserved in this way are dependent on the particular clock generator used by the adapter.

The driver currently supports all programmable clock generators known to exist on Mach64 adapters. In this case, the driver will completely ignore any XF86Config clock specification, and programme the clock generator as needed by the modes used during the X session.

5.2.2. Clocks for unsupported programmable clock generators

This case is unlikely to occur, but is documented for the sake of completeness.

In this situation, the driver will probe the adapter for clock frequencies unless XF86Config clocks are already specified. In either case, the driver will then attempt to normalize the clocks to one of the following specifications:

```
BIOS setting 1:

  Clocks   0.000 110.000 126.000 135.000  50.350  56.640  63.000  72.000
           0.000  80.000  75.000  65.000  40.000  44.900  49.500  50.000
           0.000  55.000  63.000  67.500  25.180  28.320  31.500  36.000
           0.000  40.000  37.500  32.500  20.000  22.450  24.750  25.000

BIOS setting 2:

  Clocks   0.000 110.000 126.000 135.000  25.180  28.320  31.500  36.000
           0.000  80.000  75.000  65.000  40.000  44.900  49.500  50.000
           0.000  55.000  63.000  67.500  12.590  14.160  15.750  18.000
           0.000  40.000  37.500  32.500  20.000  22.450  24.750  25.000

BIOS setting 3:

  Clocks   0.000   0.000   0.000   0.000  25.180  28.320   0.000   0.000
           0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
           0.000   0.000   0.000   0.000  12.590  14.160   0.000   0.000
           0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
```

If the driver matches the clocks to the third setting above, functionality will be *extremely* limited (assuming the driver works at all).

5.2.3. Clocks for fixed clock generators on ATI adapters

This section applies to all ATI adapters except all but the very earliest Mach64's.

One of the following clocks specifications (or an initial subset thereof) can be used depending on what the adapter uses to generate dot clocks:

```
Crystals (VGA Wonder V3 and V4 adapters only):
```


```
  Clocks  50.000  56.644   0.000  44.900  44.900  50.000   0.000  36.
          25.000  28.322   0.000  22.450  22.450  25.000   0.000  18.
          16.667  18.881   0.000  14.967  14.967  16.667   0.000  12.000
          12.500  14.161   0.000  11.225  11.225  12.500   0.000   9.000

ATI 18810 clock generator:

  Clocks  30.240  32.000  37.500  39.000  42.954  48.771   0.000  36.000
          40.000  56.644  75.000  65.000  50.350  56.640   0.000  44.900
          15.120  16.000  18.750  19.500  21.477  24.386   0.000  18.000
          20.000  28.322  37.500  32.500  25.175  28.320   0.000  22.450
          10.080  10.667  12.500  13.000  14.318  16.257   0.000  12.000
          13.333  18.881  25.000  21.667  16.783  18.880   0.000  14.967
           7.560   8.000   9.375   9.750  10.739  12.193   0.000   9.000
          10.000  14.161  18.750  16.250  12.586  14.160   0.000  11.225

ATI 18811-0 and ATI 18812-0 clock generators:

  Clocks  30.240  32.000 110.000  80.000  42.954  48.771  92.400  36.000
          39.910  44.900  75.000  65.000  50.350  56.640   0.000  44.900
          15.120  16.000  55.000  40.000  21.477  24.386  46.200  18.000
          19.955  22.450  37.500  32.500  25.175  28.320   0.000  22.450
          10.080  10.667  36.667  26.667  14.318  16.257  30.800  12.000
          13.303  14.967  25.000  21.667  16.783  18.880   0.000  14.967
           7.560   8.000  27.500  20.000  10.739  12.193  23.100   9.000
           9.978  11.225  18.750  16.250  12.588  14.160   0.000  11.225

ATI 18811-1 and ATI 18811-2 clock generators:

  Clocks 135.000  32.000 110.000  80.000 100.000 126.000  92.400  36.000
          39.910  44.900  75.000  65.000  50.350  56.640   0.000  44.900
          67.500  16.000  55.000  40.000  50.000  63.000  46.200  18.000
          19.955  22.450  37.500  32.500  25.175  28.320   0.000  22.450
          45.000  10.667  36.667  26.667  33.333  42.000  30.800  12.000
          13.303  14.967  25.000  21.667  16.783  18.880   0.000  14.967
          33.750   8.000  27.500  20.000  25.000  31.500  23.100   9.000
           9.978  11.225  18.750  16.250  12.588  14.160   0.000  11.225
```

VGAWonder VLB, VGA 1024 VLB, Mach32 and Mach64 owners should only specify up to the first 32 frequencies.

Other clock generators that have been used on ATI adapters (which can all be said to be clones of one of the above) might generate non-zero frequencies for those that are zero above, or vice-versa.

The order of the clocks *is* very important, although the driver will reorder the clocks if it deems it appropriate to do so. Mach32 and Mach64 owners should note that this order is different than what they would use for the accelerated servers.

5.2.4. Clocks for non-ATI adapters

If no clocks are specified in the XF86Config, the driver will probe for four clocks, the second of which will be assumed to be 28.322MHz. You can include up to four clock frequencies in your XF86Config to specify the actual values used by the adapter. Any more will be ignored.

5.3. Option "nolinear"

By default, the driver will enable a linear video memory aperture for 256-colour and higher depth modes if it is also using a Mach64 accelerator CRTC or an integrated Mach64 graphics chip. This option disables this linear aperture. Currently, this also disables support for more than 256 colours.

5.4. MemBase address

This specification is only effective for non-PCI Mach64 adapters, and is used to override the CPU address at which the adapter will map its video memory. Normally, for non-PCI adapters, this address is set by a DOS install utility provided with the adapter. The MemBase option can also be used to enable the linear aperture in those cases where ATI's utility was not, or can not be, used.

For PCI adapters, this address is determined at system bootup according to the PCI Plug'n'Play specification which arbitrates the resource requirements of most devices in the system. This means the driver can not easily change the linear aperture address.

5.5. Modelines

Modes can be derived from the information in XFree86's doc directory. If you do not specify a "modes" line in the display subsection of the appropriate screen section of your XF86Config, the driver will generate a default mode and attempt to use it. The timings for the default mode are derived from the timings of the mode (usually a text mode) in effect when the server is started.

ATI Adapters README file **:** *XF86Config specifications*
Previous: *Current implementation of generic VGA support for non-ATI adapters*
Next: *Known problems and limitations*

6. Known problems and limitations

There are several known problems or limitations related to the XFree86 ATI driver. They include:

- A number of system lockups and blank screens have been reported when using PCI Mach64 adapters. The great majority of these problems have been found to be due to system aspects that are unrelated to this driver. As of this writing, these problems can be divided into three general areas:

  Improper mouse protocol specification with some recent mice. Try different protocol specifications or another mouse.

  A system conflict with APM. This problem is Linux-specific. There is a bug in kernels 2.0.31 or earlier that prevents proper APM operation. Upgrade to a more recent kernel or disable APM support.

  The TV port on some Mach64 adapters needs to be disabled using an ATI utility that might or might not be supplied with the adapter. This problem is currently under investigation.
- When using a Mach64's accelerator CRTC, the virtual resolution must be less than 8192 pixels wide. The VGA CRTC further limits the virtual resolution width to less than 4096 pixels, or to less than 2048 pixels for adapters based on 18800's (with 256kB of memory) and on Mach64 integrated controllers. These are hardware limits that cannot be circumvented.
- Virtual resolutions requiring more than 1MB of video memory (256kB in the monochrome case) are not supported by the VGA CRTC on 88800GX and 88800CX adapters. This is a hardware limit that cannot be circumvented.
- Due to hardware limitations, doublescanned modes are not supported by the accelerator CRTC in 88800GX, 88800CX, 264CT and 264ET adapters.
- Monochrome interlaced modes are not supported on 18800-x and 28800-x when using a virtual resolution that is 2048 pixels or wider. This is yet another hardware limitation that cannot be circumvented.
- Video memory banking does not work in monochrome and 16-colour modes on 18800 and 18800-1 adapters. This appears to be another hardware limit, but this conclusion cannot be confirmed at this time. The driver's default behaviour in this case is to limit video memory to 256kB.
- The default mode does not work on the more recent Mach64 adapters. This problem is caused by the driver's attempt to use an incorrect dot clock for the mode. This will be fixed in a future release by reading the programmable clock generator's registers to determine the actual clock used by the mode.
- Most XFree86 servers assume that the video state on entry to the server is a text mode. This assumption is known to cause problems on operating systems which invoke the server from a

graphics mode. DBCS versions of OS/2, primarily used in Asia, are examples of such operating systems. The solution, for now, is to somehow coerce the OS to invoke the server from a text mode. This driver has been changed to simply assume the mode on entry uses the adapter's VGA CRTC (in text or graphics modes). While this action alleviates the problem somewhat, it does not completely solve it, as the server could still be invoked from an accelerator mode. To properly fix this problem for all XFree86 servers is a large project, and will probably not get done anytime soon.

- Video memory corruption can still occur during mode switches on 18800 and 18800-1 adapters. Symptoms of this problem include garbled fonts on return to text mode, and various effects (snow, dashed lines, etc) on initial entry into a graphics mode. In the first case, the workaround is to use some other means of restoring the text font. On Linux, this can be accomplished with the kbd or svgalib packages. In the second case, xrefresh(1) will usually clean up the image. No solution to this problem is currently known.
- There is some controversy over what the maximum allowed clock frequency should be on 264xT and 3D Rage adapters. For now, clocks will, by default, be limited to 135MHz, 170MHz, 200MHz or 230MHz, depending on the specific controller. This limit can only be increased (up to a driver-calculated absolute maximum) through the DACSpeed specification in XF86Config. Be aware however that doing so is untested and might damage the adapter.
- Except as in the previous item, clocks are limited to 80MHz on most adapters, although many are capable of higher frequencies. This will be fixed in a future release.

Support for the following will be added in a future release:

- Mach32 accelerator's CRTC. This support is the first step towards accelerated support for Mach32's, Mach8's, 8514/A's and other clones.
- Colour depth greater than 8, where permitted by the hardware.
- Mach64, Mach32, Mach8 and 8514/A Draw Engines.
- Hardware cursors.

Support, through this driver, for 3D acceleration, "TV in a window" and video capture, as implemented in some ATI adapters, is still in exploratory stages. There is currently no framework within an XFree86 server for these functions, although one is in development. Also, ATI has not yet released a register-level specification for these, except under non-disclosure agreements.

ATI Adapters README file **:** *Known problems and limitations*
Previous: *XF86Config specifications*
Next: *Reporting problems*

7. Reporting problems

If you are experiencing problems that are not already recorded in this document, first ensure that you have the latest current release of this driver and XFree86. Check the server's stderr output and ftp://ftp.xfree86.org/pub/XFree86 if you are uncertain.

Secondly, please check XFree86's doc directory for additional information.

Thirdly, do not forget to read http://www.xfree86.org/FAQ.

Fourth, a scan through the comp.windows.x.i386unix and comp.os.linux.x newsgroups using your favourite archiving service can also prove useful in resolving problems.

If you are still experiencing problems, you can send me e-mail at *tsi@ualberta.ca*. Please be as specific as possible when describing the problem(s), and include an unedited copy of the server's stderr and the XF86Config file used.

8. Driver history

The complete history of the driver is rather cloudy. The following is more than likely to be incomplete and inaccurate.

Apparently, Per Lindqvist first got a driver working with an early ATI adapter under X386 1.1a. This original driver might have actually been based on a non-functional ATI driver written by Thomas Roell (currently of Xi Graphics).

Then Doug Evans (*dje@cygnus.com*) added support for the ATI VGA Wonder XL, trying in the process to make the driver work with all other ATI adapters available at the time.

Rik Faith (*faith@cs.unc.edu*) obtained the X11R4 driver from Doug Evans in the summer of 1992 and ported the code to the X386 part of X11R5. This subsequently became part of XFree86.

I (Marc Aurele La France) took over development and maintenance of the driver in the fall of 1993 after Rik got rid of his VGA Wonder card.

9. Driver versions

Due to the introduction of loadable drivers in an upcoming XFree86 release, it has become necessary to track driver versions separately. With this release of the driver, I am introducing the following version numbering scheme.

Version 1 of this driver is the one I inherited from Rik Faith. This is the version found in XFree86 2.0 and 2.1.

Version 2 is my first rewrite of this code which only ended up being a partially unsuccessful attempt at generalizing the driver for all VGA Wonder, Mach32, and early Mach64 adapters. Various releases of this version of the driver can be found in XFree86 2.1.1, 3.1, 3.1.1 and 3.1.2.

Version 3 represents my second rewrite (although a rather lame one as rewrites go). Into version 3, I introduced clock programming for Mach64 adapters and merged in the old ati_test debugging tool. This is the version found in XFree86 3.2, 3.3 and 3.3.1.

Version 4 is a rather major restructuring of version 3, which became larger than I could comfortably handle in one source file. This version will make it quite a bit easier to introduce new function such as acceleration, additional colour depths, and so on. This is the version found in XFree86 3.3.2, 3.3.3, 3.3.3.1, 3.3.3.2 and 3.3.4.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/ati.sgml,v 3.15.2.4 1999/07/05
09:07:28 hohndel Exp $
```

```
$XConsortium: ati.sgml /main/9 1996/10/19 18:03:54 kaleb $
```

Information for Chips and Technologies Users

David Bateman (*dbateman@eng.uts.edu.au*), Egbert Eich (*Egbert.Eich@Physik.TH-Darmstadt.DE*)

5th June 1998

1. Introduction
2. Supported Chips
3. XF86Config Options
4. Modelines
5. The Full Story on Clock Limitations
6. Troubleshooting
7. Disclaimer
8. Acknowledgement
9. Authors

1. Introduction

The Chips and Technologies range of chips are primarily manufactured for use in laptop computers, where their power conservation circuitry is of importance. They can however be found in a few "`Green`" video cards for desktop machines. This release of XFree86 includes support for

- Linear Addressing
- 16/24/32 bits per pixel
- Fully programmable clocks are supported
- H/W cursor support
- BitBLT acceleration of many operations using XAA

Most of the Chips and Technologies chipsets are supported by this driver to some degree.

2. Supported Chips

ct65520

(Max Ram: 1Mb, Max Dclk: 68MHz@5V)

ct65525

This chip is basically identical to the 65530. It has the same ID and is identified as a 65530 when probed. See ct65530 for details.

ct65530

This is a very similar chip to the 65520. However it additionally has the ability for mixed 5V and 3.3V operation and linear addressing of the video memory. (Max Ram: 1Mb, Max Dclk: 56MHz@3.3V, 68MHz@5V)

ct65535

This is the first chip of the ct655xx series to support fully programmable clocks. Otherwise it has the the same properties as the 65530.

ct65540

This is the first version of the of the ct655xx that was capable of supporting Hi-Color and True-Color. It also includes a fully programmable dot clock and supports all types of flat panels. (Max Ram: 1Mb, Max Dclk: 56MHz@3.3V, 68MHz@5V)

ct65545

The chip is very similar to the 65540, with the addition of H/W cursor, pop-menu acceleration, BitBLT and support of PCI Buses. PCI version also allow all the BitBLT and H/W cursor registers to be memory mapped 2Mb above the Base Address. (Max Ram: 1Mb, Max Dclk: 56MHz@3.3V,68MHz@5V)

ct65546

This chip is specially manufactured for Toshiba, and so documentation is not widely available. It is believed that this is really just a 65545 with a higher maximum dot-clock of 80MHz. (Max Ram: 1Mb?, Max Dclk: 80MHz?)

ct65548

This chip is similar to the 65545, but it also includes XRAM support and supports the higher dot clocks of the 65546. (Max Ram: 1Mb, Max Dclk: 80MHz)

ct65550

This chip started a completely new architecture to previous ct655xx chips. It includes many new

features, including improved BitBLT support (24bpp color expansion, wider maximum pitch, Multimedia unit (video capture, zoom video port, etc) and 24bpp uncompressed true color (i.e 32bpp mode). Also memory mapped I/O is possible on all bus configurations. (Max Ram: 2Mb, Max Dclk: 80MHz@3.3V,100MHz@5V)

(Note: At least one non-PCI bus system with a ct65550 requires the "`-no_bios`" option for the SuperProbe to correctly detect the chipset with the factory default BIOS settings. The XF86_SVGA server can correctly detect the chip in the same situation.)

ct65554

This chip is similar to the 65550 but has a 64bit memory bus as opposed to a 32bit bus. It also has higher limits on the maximum memory and pixel clocks (Max Ram: 4Mb, Max Dclk: 100MHz@3.3V)

ct65555

Similar to the 65554 but has yet higher maximum memory and pixel clocks. It also includes a new DSTN dithering scheme that improves the performance of DSTN screens. (Max Ram: 4Mb, Max Dclk: 110MHz@3.3V)

ct68554

Similar to the 65555 but also incorporates "PanelLink" drivers. This serial link allows an LCD screens to be located up to 100m from the video processor. Expect to see this chip soon in LCD desktop machines (Max Ram: 4Mb, Max Dclk: 110MHz@3.3V)

ct69000

Similar to the 65555 but incorporates 2Mbytes of SGRAM on chip. It is the first Chips and Technologies chipset where all of the registers are accessible through MMIO, rather than just the BitBlt registers. (Max Ram: 2Mb Only, Max Dclk: 220MHz@3.3V)

ct64200

This chip, also known as the WinGine, is used in video cards for desktop systems. It often uses external DAC's and programmable clock chips to supply additional functionally. None of these are currently supported within the driver itself, so many cards will only have limited support. Linear addressing is not supported for this card in the driver. (Max Ram: 2Mb, Max Dclk: 80MHz)

ct64300

This is a more advanced version of the WinGine chip, with specification very similar to the 6554x series of chips. However there are many difference at a register level. A similar level of acceleration to the 65545 is included for this driver. (Max Ram: 2Mb, Max Dclk: 80MHz)

---

3. XF86Config Options

The following options are of particular interest to the Chips and Technologies driver. Each of them must be specified in the `svga' driver section of the XF86Config file, within the Screen subsections of the depths to which they are applicable (you can enable options for all depths by specifying them in the Device section).

Option "noaccel"

This option will disable the use of any accelerated functions. This is likely to help with some problems related to DRAM timing, high dot clocks, and bugs in accelerated functions, at the cost of performance (which will still be reasonable on VLB/PCI).

Option "no_bitblt" (Chips 65545 and later)

This option will disable the use of the BitBLT engine which the 65545 and above have. If you can use the "`noaccel`" option to correct a problem, then this option might be better to use. It still allows the use of generic speedups.

Option "xaa_no_color_exp" (Chips 65545 and later)

This option will have the effect of disabling the use of monochrome colour expansion. In particular this effects text and bitmaps. It is useful for problems related to image writes, and possible acceleration problems. In general this will result in a reduced performance. Note that this option replaces the "`no_imageblt`" option used in XFree86 3.2.

Option "xaa_benchmark" (Chips 65545 and later)

Turns on the XAA acceleration benchmarks. Information regarding what graphics primitives are accelerated and their relatives speeds will be printed when the X server starts.

videoram 1024 (or another value)

This option will override the detected amount of video memory, and pretend the given amount of memory is present on the card. Note that many ct655xx chips only allow up to 1Mb of videoram, and the amount should be correctly detected.

Option "nolinear" (Chips 65530 and later)

By default linear addressing is used on all ct655xx chips. However this might be broken in some implementations. It is possible to turn the linear addressing off with this option. Note that H/W acceleration and 16/24/32bpp are only supported with linear addressing.

MemBase 0x03b00000 (or a different address)

This sets the physical memory base address of the linear framebuffer. Typically this is probed correctly, but if you believe it to be mis-probed, this option might help. Also for non PCI machines specifying this force the linear base address to be this value, reprogramming the video processor to

suit. Note that for the 65530 this is required as the base address can't be correctly probed.

Option "hw_cursor" (Chips 65545 and later)

This option enables the use of a hardware accelerated cursor. This effectively speeds all graphics operations as the job of ensuring that the cursor remains on top is now given to the hardware. It also reduces the effect of cursor flashing during graphics operations.

Option "sw_cursor" (Chips 65545 and later)

Software cursors have now been made the default and so this option has no effect.

Option "STN"

The server is unable to differentiate between SS STN and TFT displays. This forces it to identify the display as a SS STN rather than a TFT.

Option "use_modeline"

The flat panel timings are related to the panel size and not the size of the mode specified in XF86Config. For this reason the default behaviour of the server is to use the panel timings already installed in the chip. The user can force the panel timings to be recalculated from the modeline with this option. However the panel size will still be probed.

Option "fix_panel_size"

For some machines the LCD panel size is incorrectly probed from the registers. This option forces the LCD panel size to be overridden by the modeline display sizes. This will prevent the use of a mode that is a different size than the panel. Before using this check that the server reports an incorrect panel size. This option can be used in conjunction with the option "use_modeline" to program all the panel timings using the modeline values.

Option "no_stretch"

When the size of the mode used is less than the panel size, the default behaviour of the server is to stretch the mode in an attempt to fill the screen. A "`letterbox`" effect with no stretching can be achieved using this option.

Option "lcd_center"

When the size of the mode used is less than the panel size, the default behaviour of the server is to align the left hand edge of the display with the left hand edge of the screen. Using this option the mode can be centered in the screen. This option is reported to have problems with some machines at 16/24/32bpp, the effect of which is that the right-hand edge of the mode will be pushed off the screen.

Option "hw_clocks" (Chips 65535 and later)

On chips 65535 and later, the default is to use the programmable clock for all clocks. It is possible to use the fixed clocks supported by the chip instead by using this option. Typically this will give you some or all of the clocks 25.175, 28.322, 31.000 and 36.000MHz. The current programmable clock will be given as the last clock in the list. On a cold-booted system this might be the appropriate value to use at the text console (see the "`TextClockFreq`" option), as many flat panels will need a dot clock different than the default to synchronise. The programmable clock

makes this option obsolete and so it's use isn't recommended.

Option "use_vclk1" (Chips 65550 and later)

The HiQV series of chips have three programmable clocks. The first two are usually loaded with 25.175 and 28.322MHz for VGA backward compatibility, and the third is used as a fully programmable clock. On at least one system (the Inside 686 LCD/S single board computer) the third clock is unusable. This option forces the use of VClk1 as the programmable clock. It has been reported that this option can fix the blue/black screen problem on startup that a few machines suffer.

TextClockFreq 25.175

It is impossible for the server to read the value of the currently used frequency for the text console from the chip with the ct6554x series of chips. Therefore the server uses a default value of 25.175MHz as the text console clock. For some LCDs, in particular DSTN screens, this clock will be wrong. This allows the user to select a different clock for the server to use when returning to the text console.

Option "mmio"

This enables the use of memory-mapped I/O to talk to the BitBLT engine. By default memory-mapped I/O is not enabled on the 6554x series of chips, and is only usable on 6554x's with PCI buses. This option has no effect when not using the BitBLT engine (e.g. when using "`no_bitblt`"), or for the 65550 which can only use MMIO for access to the BitBLT engine. On 65545 PCI machines MMIO is enabled by default because the blitter can not be used otherwise.

Option "suspend_hack"

This option sets the centering and stretching to the bios default values. This can fix suspend/resume problems on some machines. It overrides the options "lcd_center" and "no_stretch".

Option "use_18bit_bus" (Chips 65540/45/46/48)

For 24bpp on TFT screens, the server assumes that a 24bit bus is being used. This can result in a reddish tint to 24bpp mode. This option, selects an 18 bit TFT bus. For other depths this option has no effect.

Chipset "ct65546" (or some other chip)

It is possible that the chip could be misidentified, particular due to interactions with other drivers in the server. It is possible to force the server to identify a particular chip with this option.

Option "sync_on_green" (Chips 65550/54/55 and 68554)

Composite sync on green. Possibly useful if you wish to use an old workstation monitor. The 65550/54 internal RAMDAC's support this mode of operation, but whether a particular machine does depends on the manufacturer.

Option "fast_dram" (Chips 65550/54/55 and 68554)

This option sets the internal memory clock (MCLK) registers to 38MHz. The default value programmed by the BIOS is usually OK, but some machines can accept a faster MClk to achieve a

better performance. One machine known to work well with this option is the Toshiba 720CDT. Note that newer machines often have an MClk greater than 38MHz, and so this option might actually slower the machine down. This option is generally not recommended and is superseded by the "`Set_MemClk`" option.

DacSpeed 80.000

The server will limit the maximum dotclock to a value as specified by the manufacturer. This might make certain modes impossible to obtain with a reasonable refresh rate. Using this option the user can override the maximum dot-clock and specify any value they prefer. Use caution with this option, as driving the video processor beyond its specifications might cause damage.

Set_MemClk 38.000 (Chips 65550/54/55 and 68554)

This option sets the internal memory clock (MCLK) registers to 38MHz or some other value. Use caution as excess heat generated by the video processor if its specifications are exceeded might cause damage. However careful use of this option might boost performance.

Information for Chips and Technologies Users **:** *XF86Config Options*
Previous: *Supported Chips*
Next: *Modelines*

4. Modelines

When constructing a modeline for use with the Chips and Technologies driver you'll needed to considered several points

*** Virtual Screen Size**

It is the virtual screen size that determines the amount of memory used by a mode. So if you have a virtual screen size set to 1024x768 using a 800x600 at 8bpp, you use 768kB for the mode. Further to this some of the XAA acceleration requires that the display pitch is a multiple of 64 pixels. So the driver will attempt to round-up the virtual X dimension to a multiple of 64, but leave the virtual resolution untouched. This might further reduce the available memory.

*** 16/24/32 Bits Per Pixel**

Chips later than the ct65540 are capable of supporting Hi-Color and True-Color modes. These are implemented in the current server. The clocks in the 6554x series of chips are internally divided by 2 for 16bpp and 3 for 24bpp, allowing one modeline to be used at all depths. The effect of this is that the maximum dot clock visible to the user is a half or a third of the value at 8bpp. The 6555x series of chips doesn't need to use additional clock cycles to display higher depths, and so the same modeline can be used at all depths, without needing to divide the clocks. Also 16/24/32 bpp modes will need 2 , 3 or 4 times respectively more video ram.

*** Frame Acceleration**

Many DSTN screens use frame acceleration to improve the performance of the screen. This can be done by using an external frame buffer, or incorporating the framebuffer at the top of video ram depending on the particular implementation. The Xserver assumes that the framebuffer, if used, will be at the top of video ram. The amount of ram required for the framebuffer will vary depending on the size of the screen, and will reduce the amount of video ram available to the modes. Typical values for the size of the framebuffer will be 61440 bytes (640x480 panel), 96000 bytes (800x600 panel) and 157287 bytes (1024x768 panel).

*** H/W Acceleration**

The H/W cursor will need 1kB for the 6554x and 4kb for the 65550. On the 64300 chips the H/W cursors is stored in registers and so no allowance is needed for the H/W cursor. In addition to this many graphics operations are speeded up using a "`pixmap cache`". Leaving too little memory available for the cache will only have a detrimental effect on the graphics performance.

*** VESA like modes**

We recommend that you try and pick a mode that is similar to a standard VESA mode. If you don't a suspend/resume or LCD/CRT switch might mess up the screen. This is a problem with the video BIOS not knowing about all the funny modes that might be selected.

*** Dot Clock**

For LCD screens, the lowest clock that gives acceptable contrast and flicker is usually the best one. This also gives more memory bandwidth for use in the drawing operations. Some users prefer to use clocks that are defined by their BIOS. This has the advantage that the BIOS will probably restore the clock they specified after a suspend/resume or LCD/CRT switch. For a complete discussion on the dot clock limitations, see the

next section.

The driver is capable of driving both a CRT and a flat panel display. In fact the timing for the flat panel are dependent on the specification of the panel itself and are independent of the particular mode chosen. For this reason it is recommended to use one of the programs that automatically generate XF86Config files, such as "`xf86config`" or "`XF86Setup`".

However there are many machines, particular those with 800x600 screen or larger, that need to reprogram the panel timings. The reason for this is that the manufacturer has used the panel timings to get a standard EGA mode to work on flat panel, and these same timings don't work for an SVGA mode. For these machines the "`use_modeline`" and/or possibly the "`fix_panel_size`" option might be needed. Some machines that are known to need these options include.

```
Modeline "640x480@8bpp"   25.175  640  672  728  816   480  489  501  526
Modeline "640x480@16bpp"  25.175  640  672  728  816   480  489  501  526
Options: "use_modeline"
Tested on a Prostar 8200, (640x480, 65548, 1Mbyte)

Modeline "800x600@8bpp"   28.322  800  808  848  936   600  600  604  628
Options: "fix_panel_size", "use_modeline"
Tested on a HP OmniBook 5000CTS (800x600 TFT, 65548, 1Mbyte)

Modeline "800x600@8bpp"   30.150  800  896  960 1056   600  600  604  628
Options: "fix_panel_size", "use_modeline"
Test on a Zeos Meridan 850c (800x600 DSTN, 65545, 1Mbyte)
```

The NEC Versa 4080 just needs the "fix_panel_size" option.

Information for Chips and Technologies Users : *Modelines*
Previous: *XF86Config Options*
Next: *The Full Story on Clock Limitations*

5. The Full Story on Clock Limitations

There has been much confusion about exactly what the clock limitations of the Chips and Technologies chipsets are. Hence I hope that this section will clear up the misunderstandings.

In general there are two factors determining the maximum dotclock. There is the limit of the maximum dotclock the video processor can handle, and there is another limitation of the available memory bandwidth. The memory bandwidth is determined by the clock used for the video memory. For chipsets incapable of colour depths greater that 8bpp like the 65535, the dotclock limit is solely determined by the highest dotclock the video processor is capable of handling. So this limit will be either 56MHz or 68MHz for the 655xx chipsets, depending on what voltage they are driven with, or 80MHz for the 64200 WinGine machines.

The 6554x and 64300 WinGine chipsets are capable of colour depths of 16 or 24bpp. However there is no reliable way of probing the memory clock used in these chipsets, and so a conservative limit must be taken for the dotclock limit. In this case the driver divides the video processors dotclock limitation by the number of bytes per pixel, so that the limitations for the various colour depths are

	8bpp	16bpp	24bpp
64300	85	42.5	28.33
65540/65545 3.3v	56	28	18.67
65540/65545 5v	68	34	22.67
65546/65548	80	40	26.67

For a CRT or TFT screen these limitations are conservative and the user might safely override them with the "`DacSpeed`" option to some extent. However these numbers take no account of the extra bandwidth needed for DSTN screens.

For the HiQV series of chips, the memory clock can be successfully probed. Hence you will see a line like

```
(--) SVGA: CHIPS: probed memory clock of 40090 KHz
```

in your startx log file. Note that many chips are capable of higher memory clocks than actually set by BIOS. You can use the Set_MClk option in your XF86Config file to get a higher MClk. However some video ram, particularly EDO, might not be fast enough to handle this, resulting in drawing errors on the screen. The formula to determine the maximum usable dotclock on the HiQV series of chips is

```
Max Dotclock = min(MaxDClk,  0.70  * 4 * MemoryClk / (BytesPerPixel +
                (isDSTN == TRUE ? 1 : 0)))
```

which says that there are two limits on the dotclock. One the overall maximum, and another due to the available memory bandwidth of the chip. For the memory bandwidth 4 bytes are transfered every clock cycle (Hence the 4), but after accounting for the RAS/CAS signaling only about 70% of the bandwidth is available. The whole thing is divided by the bytes per pixel, plus an extra byte if you are using a DSTN. The extra byte with DSTN screens is used for the frame buffering/acceleration in these screens. So for the various Chips and Technologies chips the maximum specifications are

	Max DClk MHz	Max Mem Clk MHz
65550 rev A 3.3v	80	38
65550 rev A 5v	110	38
65550 rev B	95	50
65554	94.5	55
65555	110	55
68554	110	55
69000	220	100

Note that all of the chips except the 65550 rev A are 3.3v only. Which is the reason for the drop in the dot clock. Now the maximum memory clock is just the maximum supported by the video processor, not the maximum supported by the video memory. So the value actually used for the memory clock might be significantly less than this maximum value. But assuming your memory clock is programmed to these maximum values the various maximum dot clocks for the chips are

	CRT/TFT			DSTN		
	8bpp	16bpp	24bpp	8bpp	16bpp	24bpp
65550 rev A 3.3v	80	53.2	35.47	53.2	35.47	26.6
65550 rev A 5v	106.2	53.2	35.47	53.2	35.47	26.6
65550 rev B	95	70	46.67	70	46.67	35.0
65554	94.5	77	51.33	77	51.33	38.5
65555	110	77	51.33	77	51.33	38.5
68554	110	77	51.33	77	51.33	38.5
69000	220	140	93.33	140	93.33	70.0

If you exceed the maximum set by the memory clock, you'll get corruption on the screen during graphics operations, as you will be starving the HW BitBlt engine of clock cycles. If you are driving the video memory too fast (too high a MemClk) you'll get pixel corruption as the data actually written to the video memory is corrupted by driving the memory too fast. You can probably get away with exceeding the Max DClk at 8bpp on TFT's or CRT's by up to 10% or so without problems, it will just generate more heat, since the 8bpp clocks aren't limited by the available memory bandwidth.

If you find you truly can't achieve the mode you are after with the default clock limitations, look at the options "`DacSpeed`" and "`Set_MemClk`". Using these should give you all the capabilities you'll need in the server to get a particular mode to work. However use caution with these options, because there is no guarantee that driving the video processor beyond it capabilities won't cause damage.

Next: *Troubleshooting*

TOC

6. Troubleshooting

The cursor appears as a white box, after switching modes

There is a known bug in the H/W cursor, that sometimes causes the cursor to be redrawn as a white box, when the mode is changed. This can be fixed by moving the cursor to a different region, switching to the console and back again, or if it is too annoying the H/W cursor can be disabled by removing the "`hw_cursor`" option.

The cursor hot-spot isn't at the same point as the cursor

With modes on the 6555x machines that are stretched to fill the flat panel, the H/W cursor is not correspondingly stretched. This is a small and long-standing bug in the current server. You can avoid this by either using the "`no_stretch`" option or removing the `hw_cursor`" option.

The lower part of the screen is corrupted

Many DSTN screens use the top of video ram to implement a frame accelerator. This reduces the amount of video ram available to the modes. The server doesn't prevent the user from specifying a mode that will use this memory, it prints a warning on the console. The effect of this problem will be that the lower part of the screen will reside in the same memory as the frame accelerator and will therefore be corrupt. Try reducing the amount of memory consumed by the mode.

There is a video signal, but the screen doesn't sync.

You are using a mode that your screen cannot handle. If it is a non-standard mode, maybe you need to tweak the timings a bit. If it is a standard mode and frequency that your screen should be able to handle, try to find different timings for a similar mode and frequency combination. For LCD modes, it is possible that your LCD panel requires different panel timings at the text console than with a graphics mode. In this case you will need the "`use_modeline`" and perhaps also the "`fix_panel_size`" options to reprogram the LCD panel timings to sensible values.

`Wavy' screen.

Horizontal waving or jittering of the whole screen, continuously (independent from drawing operations). You are probably using a dot clock that is too high (or too low); it is also possible that there is interference with a close MCLK. Try a lower dot clock. For CRT's you can also try to tweak the mode timings; try increasing the second horizontal value somewhat.

Crash or hang after start-up (probably with a black screen).

Try the "`noaccel`" or "`no_bitblt`" options. Check that the BIOS settings are OK; in particular, disable caching of 0xa0000-0xaffff. Disabling hidden DRAM refresh may also help.

Hang as the first text is appearing on the screen on SVR4 machines.

This problem has been reported under UnixWare 1.x, but not tracked down. It doesn't occur under UnixWare 2.x and only occurs on the HiQV series of chips. It might affect some other SVR4 operating systems as well. The workaround is to turn off the use of CPU to screen acceleration with the "`xaa_no_color_exp`" option.

Crash, hang, or trash on the screen after a graphics operation.

This may be related to a bug in one of the accelerated functions, or a problem with the BitBLT engine. Try the "`noaccel`" or "`no_bitblt`" options. Also check the BIOS settings. It is also possible that with a high dot clock and depth on a large screen there is very little bandwidth left for using the BitBLT engine. Try reducing the clock.

Chipset is not detected.

Try forcing the chipset to a type that is most similar to what you have.

The screen is blank when starting X

One possible cause of this problem is if the kernel has been compiled with the "APM_DISPLAY_BLANK" option. It appears that this option doesn't work as specified and can cause the Xserver to blank when starting X. In all cases the kernel should be compiled without this option. If the problem remains a CRT/LCD or switch to and from the virtual console will often fix it.

Textmode is not properly restored

This has been reported on some configurations. Many laptops use the programmable clock of the 6554x chips at the console. It is not always possible to find out the setting that is used for this clock if BIOS has written the MClk after the VClk. Hence the server assumes a 25.175MHz clock at the console. This is correct for most modes, but can cause some problems. Usually this is fixed by switching between the LCD and CRT. Alternatively the user can use the "`TextClockFreq`" option described above to select a different clock for the text console. Another possible cause of this problem is if the kernel is compiled with the "APM_DISPLAY_BLANK" option. As mentioned before, this option should be disabled.

I can't display 640x480 on my 800x600 LCD

The problem here is that the flat panel needs timings that are related to the panel size, and not the mode size. There is no facility in the current Xservers to specify these values, and so the server attempts to read the panel size from the chip. If the user has used the "`use_modeline`" or "`fix_panel_size`" options the panel timings are derived from the mode, which can be different than the panel size. Try deleting theses options from XF86Config or using an LCD/CRT switch.

I can't get a 320x240 mode to occupy the whole 640x480 LCD

There is a bug in the 6554x's H/W cursor for modes that are doubled vertically. The lower half of the screen is not accessible. The servers solution to this problem is not to do doubling vertically. Which results in the 320x240 mode only expanded to 640x360. If this is a problem, a work around is to remove the "`hw_cursor`" option. The server will then allow the mode to occupy the whole 640x480 LCD.

After a suspend/resume my screen is messed up

During a suspend/resume, the BIOS controls what is read and written back to the registers. If the screen is using a mode that BIOS doesn't know about, then there is no guarantee that it will be resumed correctly. For this reason a mode that is as close to VESA like as possible should be selected. It is also possible that the VGA palette can be affected by a suspend/resume. Using an 8bpp, the colour will then be displayed incorrectly. This shouldn't affect higher depths, and is fixable with a switch to the virtual console and back.

The right hand edge of the mode isn't visible on the LCD

This is usually due to a problem with the "`lcd-center`" option. If this option is removed form XF86Config, then the problem might go away. Alternatively the manufacturer could have incorrectly

programmed the panel size in the EGA console mode. The "`fix_panel_size`" can be used to fo the modeline values into the panel size registers. Two machines that are known to have this problem the "`HP OmniBook 5000`" and the "`NEC Versa 4080`".

My TFT screen has a reddish tint in 24bpp mode

The server assumes that the TFT bus width is 24bits. If this is not true then the screen will appear to have a reddish tint. This can be fixed by using the "`use_18bit_bus`" option. Note that the reverse is also true. If the "`use_18bit_bus`" is used and the TFT bus width is 24bpp, then the screen will appear reddish. Note that this option only has an effect on TFT screens.

I can't start X-windows with 16, 24 or 32bpp

Firstly, is your machine capable of 16/24/32bpp with the mode specified. Many LCD displays are incapable of using a 24bpp mode. Also you need at least a 65540 to use 16/24bpp and at least a 65550 for 32bpp. The amount of memory used by the mode will be doubled/tripled/quadrupled. The correct options to start the server with these modes are

```
startx -- -bpp 16                5-6-5 RGB ('64K color', XGA)
startx -- -bpp 16 -weight 555    5-5-5 RGB ('Hicolor')
startx -- -bpp 24                8-8-8 RGB truecolor
```

or with the HiQV series of chips (6555x, 68554 or 69000) you might try

```
startx -- -bpp 32                8-8-8 RGB truecolor
```

A general problem with the server that can manifested in many way such as drawing errors, wavy screens, etc is related to the programmable clock. Many potential programmable clock register setting are unstable. However luckily there are many different clock register setting that can give the same or very similar clocks. The clock code can be fooled into giving a different and perhaps more stable clock by simply changing the clock value slightly. For example 65.00MHz might be unstable while 65.10MHz is not. So for unexplained problems not addressed above, please try to alter the clock you are using slightly, say in steps of 0.05MHz and see if the problem goes away. Alternatively, using the "`use_vclk1`" option with chips later than the 65550 might also help.

For other screen drawing related problems, try the "`noaccel`" or "`no_bitblt`" options. A useful trick for all laptop computers is to switch between LCD/CRT (usually with something like Fn-F5), if the screen is having problems.

If you are having driver-related problems that are not addressed by this document, or if you have found bugs in accelerated functions, you can try contacting the XFree86 team (the current driver maintainer can be reached at *dbateman@eng.uts.edu.au* or *Egbert.Eich@Physik.TH-Darmstadt.DE)*, or post in the Usenet newsgroup "*comp.windows.x.i386unix*".

Information for Chips and Technologies Users **:** *Troubleshooting*
Previous: *The Full Story on Clock Limitations*
Next: *Disclaimer*

7. Disclaimer

XFree86, allows the user to do damage to their hardware with software. Although the authors of this software have tried to prevent this, they disclaim all responsibility for any damage caused by the software. Use caution, if you think the Xserver is frying your screen, TURN THE COMPUTER OFF!!

8. Acknowledgement

The authors of this software wish to acknowledge the support supplied by Chips and Technologies during the development of this software.

Information for Chips and Technologies Users **:** *Acknowledgement*
Previous: *Disclaimer*
Next: *Authors*

9. Authors

`Major Contributors` (In no particular order)

- Nozomi Ytow
- Egbert Eich
- David Bateman
- Xavier Ducoin

`Contributors` (In no particular order)

- Ken Raeburn
- Shigehiro Nomura
- Marc de Courville
- Adam Sulmicki
- Jens Maurer

We also thank the many people on the net who have contributed by reporting bugs and extensively testing this server before its inclusion in XFree86 3.2

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/chips.sgml,v 3.12.2.12 1998/11/07
13:52:45 dawes Exp $
```

Information for Cirrus Chipset Users

Harm Hanemaayer (*H.Hanemaayer@inter.nl.net*), Randy Hendry (*randy@sgi.com*) (64xx), Corin Anderson (*corina@the4cs.com*)

5 November 1998

1. Supported chipsets

2. Basic configuration

3. XF86Config options

4. Mode issues

5. Linear addressing and 16bpp/24bpp/32bpp modes

6. The ``cl64xx'' Driver

7. Trouble shooting with the ``cirrus'' driver

8. Tested Configurations

9. Driver Changes

---

1. Supported chipsets

There are two different SVGA drivers for Cirrus chipsets, one called ``cirrus'' and one called ``cl64xx''. The ``cirrus'' driver is used in the 256-color SVGA server (with acceleration) and the mono server (without acceleration). The SVGA server supports 16, 24, and 32 bits-per-pixel truecolor modes on some configurations. The ``cl64xx'' driver is used in the 256-color SVGA, 16-color and mono servers. Note that except where stated otherwise, this document is referring to the ``cirrus'' driver. The following chipsets by Cirrus Logic are supported:

CL-GD5420

ISA SVGA chipset, 1Mbyte; maximum dot clock is 45 MHz (256 color server). Acceleration with extended write modes (used for scrolling and solid filling in this driver). This chipset can *not* support 1024x768 non-interlaced in 256 colors.

CL-GD5422

Enhanced version of the 5420 (32-bit internal memory interface). Maximum dot clock is 80 MHz.

CL-GD6205/6215/6225/6235

Laptop chipsets more or less compatible with the 5420. The only dot clock supported is 25 MHz (more on an external display). Some problems have been reported with these chipsets (especially on external displays). Take note of the "`noaccel`" option.

CL-GD6420/6440

These chipsets are not compatible with the 542x series, but are supported by the ``cl64xx'' driver. It is used in recent laptops, and bears some similarity to old Cirrus chipsets (5410/AVGA2). The driver may also work for other 64xx chips. The configuration identifiers for this driver are "`cl6420`" and "`cl6440`". This driver is discussed in detail in section The cl64xx Driver.

CL-GD5424

Basically VLB version of the 5422, but resembles the 5426 in some respects.

CL-GD5426

Supports both ISA bus and VLB, and up to 2Mbyte of memory. Has BitBLT engine for improved acceleration (BitBlt, image transfer, text). Dot clock limit is 85 MHz.

CL-GD5428

Enhanced version of the 5426.

CL-GD5429

Enhanced version of the 5428; officially supports higher MCLK and has memory-mapped I/O.

CL-GD5430

Similar to 5429, but with 543x core (32-bit host interface). Does not have 64-bit memory mode.

CL-GD5434

`Alpine' family chip with 64-bit internal memory interface. The chip can only support 64-bit mode if equipped with 2 Mbytes of memory; cards with only 1 Mbyte are severely limited. Supports dot clocks up to 110 MHz (later chips support 135 MHz).

CL-GD5436


TOC

Highly optimized 5434.

CL-GD5440

Similar to the CL-GD5430, and detected as such.

CL-GD5446

Another member of the Alpine family of 2D accelerators; similar to the CL-GD5436.

CL-GD5480

Newer Alpine family chip that support synchronous graphics RAM (SGRAM).

CL-GD5462, CL-GD5464 and CL-GD5465

The Laguna VisualMedia family of 2D Accelerators. These chips use Rambus RDRAM memory. The '62 is a 64-bit 2D accelerator, including a BitBlit engine, video windows (not currently used by the server), and 64x64 HW cursor. Mono modes have not been tested. The CL-GD5464 is the next chip in the Laguna family, and the CL-GD5465 is the latest member, both have been tested.

CL-GD7541/7542/7543/7548

Laptop chipsets more or less compatible with the 5428/3x. While has it been tested on some configurations, not all configuration may work correctly.

CL-GD7555

Limited untested support, without auto-detection, has been provided for this chip which is a 64-bit extension of the 754x family. Use a Chipset "clgd7555" line.

Here's a list of maximum dot clocks for each supported depth:

	mono	8 bpp (256c)	16 bpp	24 bpp	32 bpp
CL-GD62x5	45 MHz	45 MHz			
CL-GD5420	80 MHz	45 MHz (1)			
CL-GD542x/512K	80 MHz	45 MHz			
CL-GD5422/24	80 MHz	80 MHz	40 MHz	27 MHz	
CL-GD5426/28	85 MHz	85 MHz	45 MHz (2)	28 MHz	
CL-GD5429	85 MHz	85 MHz	50 MHz	28 MHz	
CL-GD5430	85 MHz	85 MHz	45 MHz (2)	28 MHz	
CL-GD5434/1Mb	85 MHz	85 MHz	42 MHz	28 MHz	
CL-GD5434/2Mb	85 MHz	110/135 MHz	85 MHz	28 MHz	45/50 MHz (2)
CL-GD5436/1Mb	85 MHz	110 MHz (3)	60 MHz (3)	40 MHz (3)	
CL-GD5436/2Mb	85 MHz	135 MHz	85 MHz	85 MHz (3)	60 MHz (3)
CL-GD5446/1Mb	85 MHz	110 MHz (3)	60 MHz (3)	40 MHz (3)	
CL-GD5446/2Mb	85 MHz	135 MHz	85 MHz	85 MHz (3)	60 MHz (3)
CL-GD5462	170 MHz	170 Mhz	170 MHz	170 MHz	135 MHz
CL-GD5464/65	170 MHz	230 Mhz	170 MHz	170 MHz	135 MHz
CL-GD5480	85 MHz	200 MHz	100 MHz	100 MHz	50 MHz
CL-GD754x	80 MHz	80 MHz	40 MHz (4)	(5)	

(1) with 512K memory.
(2) 50 MHz with high MCLK setting.
(3) Depends on memory clock.
(4) This may be too low for some chips.
(5) This depth may actually work if it is enabled and tested.

Rough virtual/physical screen resolution limits for different amounts of video memory:

	mono	8 bpp	16 bpp	24 bpp	32 bpp
256K	800x600	640x400			
512K	1152x900	800x600	640x400		
1024K	1600x1200	1152x900	800x600	680x510	
2048K	2304x1728	1600x1200	1152x900	960x720	800x600
4096K	2304x1728	2272x1704	1600x1200	1360x1020	1152x900

For 546x chips, the above table isn't quite accurate. While the virtual width may be any size, the screen *pitch* will be rounded up to the nearest value in the table below. Thus, each line on the screen will take more video memory than just what is displayed. To maximize video memory, then, choose the virtual desktop width from the table of pixel widths below:

```
8bpp:   640, 1024, 1280, 1664, 2048, 2560, 3328, 4096, 5120, 6656
16bpp:  320,  512,  640,  832, 1024, 1280, 1664, 2048, 2560, 3328
24bpp:  640, 1024, 1280, 1664, 2048, 2560, 3328, 4096, 5120, 6656
32bpp:  160,  256,  320,  416,  512,  640,  832, 1024, 1280, 1664
```

For other Cirrus chips, it's advisable to have a virtual width that is a multiple of 32 if acceleration is used. The horizontal monitor timings must be below 2048.

To run XF86_SVGA at a higher color depth, pass options to the X server as follows:

```
startx -- -bpp 16                5-6-5 RGB ('64K color', XGA)
startx -- -bpp 16 -weight 555    5-5-5 RGB ('Hicolor') (not on 5462)
startx -- -bpp 24                8-8-8 RGB truecolor
startx -- -bpp 32                8-8-8 XRGB truecolor (543X/46/6X only)
```

Information for Cirrus Chipset Users **:** *Supported chipsets*
Previous: *Information for Cirrus Chipset Users*
Next: *Basic configuration*

2. Basic configuration

It is recommended that you generate an XF86Config file using the `XF86Setup' or `xf86config' program, which should produce a working high-resolution 8bpp configuration. You may want to include mode timings in the `Monitor` section that better fit your monitor (e.g 1152x900 modes). The driver options are described in detail in the next section; here the basic options are hinted at.

For all chipsets, a `Clockchip "cirrus"` line in the `Device` section can be useful. This allows the use of any dot clocks, instead of one out of the fixed set of dot clocks supported by the driver. This is required if you want a 12.6 MHz dot clock for low-resolution modes. However, when this option used, clock frequencies be unstable leading to strange effects, so only use it if absolutely required.

For any chip with a BitBLT engine, the new XAA (XFree86 Acceleration Architecture) is used. This code is new and still in a beta stage. If graphics redrawing goes wrong, try the `"noaccel"` option; if it is using memory-mapped I/O, `"no_mmio"` might be sufficient.

In order to be able to run at a depth of 16bpp, 24bpp, or 32bpp, and to improve performance at 8bpp, linear addressing must be enabled. This is generally

In order to be able to run at a depth of 16bpp, 24bpp or 32bpp, and to improve performance at 8bpp, linear addressing must be enabled. Linear addressing is the default mode of operation on any PCI-bus configuration; use `"nolinear"` to disable it. For other bus types, it is generally possible on 543x local bus cards, and if you have less than 16Mb of system memory, on local bus 542x cards and ISA 543x cards. You must specify the `"linear"` option and possibly a `Membase` address. See the following sections for a detailed description.

Memory-mapped I/O is the default mode of operation for any Alpine family chip. For the 5429, the `"mmio"` option may be used to enable it, but it has not been tested.

Finally, if you have 546X chip, it will be on either a PCI or AGP bus. As such, there is no problem about memory mapped I/O or linear frame buffer address spaces running into system memory. The PCI spaces are mapped way up near the 4GB point. Because the mmio and linear frame buffer don't conflict at all on the system, the `"linear"`, `Membase`, and `"mmio"` options are ignored (memory mapped I/O and linear addressing are always used).

3. XF86Config options

Don't use the `Clocks' command. The clocks are fixed (i.e. not probed), and there should be no variation between cards (other than the maximum supported clock for each chipset).

The following options are of particular interest to the Cirrus driver. Each of them must be specified in the `svga' driver section of the `XF86Config` file, within the `Screen` subsections of the depths to which they are applicable (you can enable options for all depths by specifying them in the Device section).

Option "noaccel"

This option will disable the use of any accelerated functions. This is likely to help with problems related to bugs in acceleration functions, and perhaps high dot clocks and DRAM timing, at the cost of performance (which will still be reasonable on a local bus).

Option "fast_dram" "med_dram" "slow_dram" (5424/6/8/9, 543x, 5446, 546x)

These options set the internal memory clock (MCLK, or BCLK for the 546x) register to another value. The default value programmed by the BIOS is usually OK, don't mess with these options unless absolutely required.

The `"fast_dram"` option will cause the driver to set the internal memory clock (MCLK) register of the video card to a higher value (recent chips use an even higher value by default). Normally, this register is not touched but it appears that the standard CL-GD542x BIOS initializes it to a value that is somewhat on the low side (limited by the chip specification), which has a negative influence on performance of high dot clock modes. This is especially true if extended RAS timing is being used (this is indicated in the server probe). The actual speed of DRAM is not a critical factor in the determining whether this option is appropriate; one CL-GD5426-based card with 80ns DRAM using Extended RAS timing, which came with a DOS driver utility to set the MCLK to this value (0x22), seems to run stable at higher MCLK.

There are also (mainly brand name) cards whose customized BIOS does initialize to a higher non-standard value.

The `"slow_dram"` option will set the MCLK to the value used by the standard CL-GD542x BIOS (0x1c). Symptoms of a MCLK that is too high can be vertical bands of flickering pixels on the screen, erroneous pixels appearing in text, and loosing pixels in the textmode font after running X (note that very similar effects can be caused by an MCLK setting that is too low).

Upon start-up, the driver will report the value of the MCLK register (check this first), and also any changes that are made.

Typical MCLK values:

0x1c (50 MHz)

This is usually the BIOS default. It is forced by the `"slow_dram"` option.

0x1f (55 MHz)

Value used by the `"med_dram"` option. Highest value that 542x based cards seem to be able to handle with linear addressing enabled.

0x22 (60 MHz)

Value that most (Extended RAS) 542x cards seem to be able to handle, used by the `"fast_dram"` option.

The official maximum of the 542x chips is 50 MHz. The official spec. for the 5434 is also 50 MHz (0x1c) and that for the 5429 and 5430 is probably 60 MHz (0x22). Current revisions of the 5434 (E and greater) support 60 MHz MCLK in graphics modes, and the driver will program this automatically. If it causes problems, use the `"slow_dram"` option.

The driver takes the MCLK into account for clock limits that are determined by DRAM bandwidth.

For the 546x chips, the BCLK is the Rambus access clock. Typical values live in the range of 258 MHz to 300 MHz. If you have troubles, such as a black checkerboard pattern on the screen, try using the `"med_dram"` or `"slow_dram"` options.

In all cases, if you are not having any problems (performance or stability at high dot clocks), it is best not to use any of the DRAM options.

Option "no_bitblt"

This option, when used with a 5426/28/29/3x/46/6x/754x, will have the effect of disabling the use of the BitBLT engine (which the 5424 does not have), while retaining some acceleration. This will be useful for problems related to functions that use the BitBLT engine. Performance is significantly decreased.

Option "no_imageblt"

This option is now obsolete. The "xaa_no_color_exp" option has a somewhat similar effect.

chipset "clgd54xx"

Force detection of the given chipset. Useful if you have a supported chipset that is not properly detected, or if you have an unsupported chip that might be compatible with a supported one.

videoram 1024 (or another value)

This option will override the detected amount of video memory, and pretend the given amount of memory is present on the card. This is useful on cards with 2Mbyte of memory whose DRAM configuration is not compatible with the way the driver enables the upper megabyte of memory, or if the memory detection goes wrong. It must be specified in the Device section.

Option "fifo_conservative" (5424/6/8/9/3x/46/6x/754x)

This option will set the CRT FIFO threshold to a conservative value for high dot clocks (>= 65 MHz), reducing performance but hopefully alleviating problems with what can be described as flashing `streaks', `jitter' or horizontally repeated display areas on the screen (especially when a


TOC

BitBLT operation is in progress, e.g. scrolling).

Option "fifo_aggressive" (5424/6/8/9/3x/46/6x/754x)

This option will set the CRT FIFO threshold to an aggressive value; it will be the same as that used for lower dot clocks. Theoretically it improves performance at high dot clocks, but it does not help in the vast majority of cases. In some cases with 546x chips, however, this option can help reduce horizontal streaks or otherwise fix abnormal display problems (display shifted to the left, etc.).

Option "no_2mb_banksel" (542x)

This option will cause the driver not to set the `DRAM bank select' bit to enable the upper megabyte of memory on a 2Mbyte card. This should be helpful with cards equipped with 512Kx8 DRAMs, as opposed to 256Kx4/16 DRAMs, when using a virtual screen configuration that uses more than 1Mbyte of memory.

Option "probe_clocks"

This option will force probing of dot clocks on the card. This should not be necessary, since the clocks are fixed and the same for all Cirrus chipsets.

Clockchip "cirrus"

This enables programmable clocks. It must be specified in the Device section. With this option, the clocks the modes use will be automatically selected. Do not specify any Clocks line. This option makes a 12.5 MHz clock possible for a 320x200 Doublescan mode. Note that some frequencies may be unstable (resulting in a `wavy' screen). Only tried and tested frequencies (like the default clocks) are guaranteed to be stable.

Option "linear" (542x/6/8/9/3x/754x on VL-bus)

This enables linear addressing, which is the mapping of the entire framebuffer to a high address beyond system memory, so that SVGA bank switching is not necessary. It enhances performance at 256 colors, and is currently required for 16bpp, 24bpp, and 32bpp. See section 4 for details.

Option "nolinear" (542x/6/8/9/3x/754x on PCI bus)

Linear addressing is the default mode of operation on any PCI-bus chip. For these configurations, this option disables linear addressing.

Membase 0x00e00000 (or a different address) (542x/6/8/9/3x/46/754x)

This sets the physical memory base address of the linear framebuffer. It must be specified in the Device section. It is required for non-PCI linear addressing configurations.

Option "favour_bitblt" (5426 only)

This option is now obsolete.

Option "mmio" (5429, 7548)

This enables the use of memory-mapped I/O to talk to the BitBLT engine on the 543x/5429, which is a bit faster. This is option has no effect when not using the BitBLT engine (e.g. when using "no_bitblt").

Option "no_mmio" (543x/4x)

This disables the use of memory-mapped I/O to talk to the BitBLT engine on any chip for whi is the default mode of operation.

Option "sw_cursor" (542x/3x/46/6x)

This disables use of the hardware cursor provided by the chip. Try this if the cursor seems to have problems. In particular, use this when using dot clocks greater than 85 MHz on the 5434/6 since those chips don't fully support the hardware cursor at those clocks.

Option "clgd6225_lcd"

Provides a work-around for problems on the LCD screen of some 62x5 laptop chipsets with maximum white colors.

Option "no_pixmap_cache"

When XAA is used (on any BitBLT chip), this option disables the use of a pixmap cache in XAA. It could help with certain drawing bugs.

Option "xaa_no_color_exp"

When XAA is used, this option disables the use of hardware color expansion features by XAA. Again, this might help with certain drawing bugs.

Option "no_stretch" (754x)

Disable automatic stretching (horizontal and vertical expansion) of 640x480 on a 800x600 LCD.

Option "pci_retry" (546x)

Enables a performance feature for PCI based cards. When this feature is enabled, the driver code will attempt to transmit data on the PCI bus as fast as possible. For the most part, this option is safe, but may cause trouble with other PCI devices such as PCI network cards, sound cards, SCSI controllers, etc. When this option is not selected, a safer approach (polling the VGA's command queue) is taken.

Information for Cirrus Chipset Users **:** *XF86Config options*
Previous: *Basic configuration*
Next: *Mode issues*

4. Mode issues

The accelerated 256-color driver uses 16K bytes of scratch space in video memory, and the hardware cursor also uses 1K (2K on the '6X). Consequently, a 1024x1024 virtual resolution should not be used with a 1Mbyte card.

On older chips, the use of a higher dot clock frequencies has a negative effect on the performance of graphics operations, especially BitBlt, when little video memory bandwidth is left for drawing (the amount is displayed during start-up for 542x/3x/46/6x chips). For the 542x/3x chips, with default MCLK setting (0x1c) and a 32-bit memory interface, performance with a 65 MHz dot clock can be half of that with a dot clock of 25 MHz. So if you are short on memory bandwidth and experience blitting slowness, try using the lowest dot clock that is acceptable; for example, on a 14" or 15" screen 800x600 with high refresh (50 MHz dot clock) is not so bad, with a large virtual screen.

5434-based cards with 2Mbyte of memory do much better at high dot clocks; the DRAM bandwidth is basically double that of the 542x series. The 543x chips also make more efficient use of the available DRAM bandwidth. The same goes for the 544x.

5. Linear addressing and 16bpp/24bpp/32bpp modes

Currently the framebuffer code 16-bit, 24-bit, and 32-bit pixels in the SVGA server requires linear addressing. Option "linear" can be specified in a depth-specific screen section to enable linear addressing; a MemBase setting (in the device section) is probably also required (although they are both automatically selected with PCI cards, like 5446, 546x, and some 543x based cards). There are a number of different card configurations.

If you have a 542x/543x on the ISA bus, and you have 16Mb or more of system memory, linear addressing is impossible. 16bpp is out, sorry. If you have less than 14Mb of memory, you may be able to map the framebuffer at 14Mb, using `MemBase 0x00e00000'. That's five zeros after the `e'. Unfortunately many ISA cards don't support linear addressing.

If you have a 5424/26/28/29 on VESA local bus, the situation is more complicated. There are two different types of cards w.r.t. linear addressing:

- Cards that can only map in the lower 16Mb, like cards on the ISA bus. This is the case with most cards. The same restrictions apply (i.e. you must have less than 16Mb of memory).
- Cards that connect address line A26 and always map at 64Mb + 14Mb or 64Mb. In this case specify `MemBase 0x04e00000' or `MemBase 0x04000000'. This assumes you have a VLB motherboard implementation that implements A26. Alternatively the card may map to 0x2000000, and recent cards like the 5429 usually map to 0x03e00000 (62Mb).

You will probably have to rely on trial and error. If you have less than 16Mb memory, the `wrong' membase setting will result in no graphics being displayed, but you can probably exit with ctrl-alt-backspace.

If you have >= 16Mb memory, the first type of card (and even the second type with a stupid VLB motherboard) will result in a crash (probably a spontaneous hard reboot).

It may be possible to find out the type by visual inspection. If the card has a pin at A26, it is likely to map beyond 64Mb. To do this, take the card out. At the VESA local bus pins (this is the smaller strip of connector pins at the non-slot side of the card), consider the right side (this is the side of the board where all the chips are mounted). There are 45 pins here. They are numbered 1 to 45, from the inside (i.e. the one nearest to the card end is 45). Counting from the inside, the 17th pin is probably not present, then there are pins at 18-20. The 21st is A30, the 22nd is A28 and the 23rd is A26. So, if we have no pins at at 21-23, the card doesn't map beyond 64Mb. If there's only a gap of two pins at 21 and 22 (or they are both present) and there's a pin at 23, the card does probably map beyond 64Mb. If there's a little logic near that pin on the card, it's more likely.

With a 543x on the local bus things are simpler (the Cirrus Logic windows drivers use it), but it is not

quite without problems.

With a card on the PCI bus, there is a PCI configuration register that holds the framebuffer base address which is read automatically by the driver if a PCI card is detected. The `scanpci' program can read out the PCI configuration and show the base address.

On the VESA local bus, most 543x cards have a default mapping address of 64Mb, with jumper options for 2048Mb and 32Mb. This is probably described in the documentation that came with the card, or look in the MS-Windows system.ini file (something with linearaddr = <offset in megabytes>). These different settings were added by Cirrus Logic after finding that many VLB motherboard implementations don't implement different address pins. The driver assumes a default of 64Mb if MemBase isn't specified. A few examples for MemBase:

```
MemBase 0x02000000      32Mb
MemBase 0x04000000      64Mb
MemBase 0x80000000      2048Mb
```

Finally, for 546X cards, you are in luck: there are no "issues" to worry about. The '6X will always use linear addressing and memory-mapped I/O, and will use the memory addresses up near 4GB. Yay for PCI!

The 16bpp and 32bpp modes are now fully accelerated, thanks to XAA. On more recent chips like the 5436/46 and the 546X, 24bpp is also fully accelerated. So although there are now up to 4 times as many bits to display, the X server shouldn't feel overly sluggish. Note also that the 24bpp and 32bpp modes are only supported on a limited set of cards, and with at least 2Mb of memory.

In the `XF86Config` `"Screen"` section, a `"Display"` subsection must be defined for each depth that you want to run, with separate Modes and virtual screen size. Example (2Mb of video memory):

```
Section "screen"
    SubSection "Display"
        Depth 8
        Virtual 1280 1024
        ViewPort 0 0
        Modes "640x480" "800x600" "1024x768"
        Option "linear"
    EndSubSection
    SubSection "Display"
        Depth 16
        Virtual 1024 992
        ViewPort 0 0
        Modes "640x480" "800x600" "1024x768"
        Option "linear"
    EndSubSection
    SubSection "Display"
        Depth 32
        Virtual 832 600
```


TOC

```
        ViewPort 0 0
        Modes "640x480" "800x600"
        Option "linear"
    EndSubSection
EndSection
```

Information for Cirrus Chipset Users **:** *Linear addressing and 16bpp/24bpp/32bpp modes*
Previous: *Mode issues*
Next: *The ``cl64xx'' Driver*

6. The ``cl64xx'' Driver

The cl64xx driver supports the cl-gd6440 found in many laptops. For example, Nan Tan Computer's NP9200, NP3600, etc., which are OEM-ed by Sager, ProStar, etc. and Texas Instruments TI4000 series are supported.

The driver works in LCD-only, CRT-only, and the chip's SimulScan mode which allows one to use both the LCD and external CRT displays simultaneously. The LCD and Simulscan modes' resolution is 640x480 while, for CRT-only, the standard VESA modes of 640x480, 600x800, and 1024x768 have been tested. Interlaced 1024x768 mode has never been debugged and does not work on the machines tested.

The chip has a documented maximum operating limit for its dot clock that is related to its core voltage. Specifically, for 5.0V the maximum dot clock is 65MHz and for 3.3V the maximum dot clock is 40MHz. The driver checks the core voltage and limits the maximum dot clock to the corresponding value. This translates to a maximum resolution of about 1024x768 at a 60Hz refresh rate. The internal frequency generator can be programmed higher than these limits and is done so during server startup when the clocks are probed which momentarily exceeding the chip's operating limit. Once a set of valid clocks is obtained, I would recommend using Clocks lines in `XF86Config`. Doing so will also decrease startup time significantly. The clocks may be obtained by running the X server -probeonly (see the XFree86 man page for more information about -probeonly).

The data book indicates that only a configuration of one megabyte of video memory is supported by the chip. This size has been directly set in the driver. If one finds a need, one should be able to override the default size in `XF86Config`. Also, with 1MB of video memory, one should be able to have a virtual screen size of e.g. 1024x1024 and this is possible in CRT-only screen mode. However, whenever the LCD is in use (LCD and SimulScan), the chip uses a portion of upper video ram for its own internal acceleration purposes. Thus, the maximum video memory available for virtual resolution in LCD modes is about 0.75MB e.g. 1024x768. If you set the virtual resolution above this, you will see what might be described as a compressed aliased band when the accelerated area is displayed.

Currently, the driver does not support switching of screen modes among LCD, CRT, and SimulScan, and, at least on the NP9200, the mode must be chosen at OS boot time (e.g. Linux's LILO) while the BIOS is still active. It should be possible to add screen mode type selection as a ModeLine flag option in `XF86Config` to allow for dynamic screen mode selection from within the X server. Finally, the driver does not currently support any of the powerdown saving features of the chip nor does it shut off the LCD's backlight on screen blank. I hope to implement all these features in future releases.

Some notes regarding the CL-GD6420:

The amount of video memory may not always be detected correctly. The driver source code includes two methods, one defined out. Better specify the amount of video memory with a VideoRam line in the

Device section. Use the standard 640x480 60 Hz standard mode timing with 25.175 MHz dot clock f
CRT or SIMulscan mode; for LCD-only operation, use the same mode timing but with a dot clock cl
16.257 MHz. Standard 56 Hz 800x600 is also supported on the CRT.

The primary contact for the cl6440 problems with ``cl64xx'' driver is Randy Hendry <*randy@sgi.com*>.

Information for Cirrus Chipset Users **:** *The ``cl64xx'' Driver*
Previous: *Linear addressing and 16bpp/24bpp/32bpp modes*
Next: *Trouble shooting with the ``cirrus'' driver*

7. Trouble shooting with the ``cirrus'' driver

First of all, make sure that the default modes selected from your `XF86Config` is supported by your monitor, i.e. make sure the horizontal sync limit is correct. It is best to start with standard 640x480x256 with a 25.175 MHz clock (by specifying a single horizontal sync of 31.5) to make sure the driver works on your configuration. The default mode used will always be the first mode listed in the modes line, with the highest dot clock listed for that resolution in the timing section.

Note that some VESA standard mode timings may give problems on some monitors (try increasing the horizontal sync pulse, i.e. the difference between the middle two horizontal timing values, or try multiples of 16 or 32 for all of the horizontal timing parameters).

There is a video signal, but the screen doesn't sync.

> You are using a mode that your monitor cannot handle. If it is a non-standard mode, maybe you need to tweak the timings a bit. If it is a standard mode and frequency that your monitor should be able to handle, try to find different timings for a similar mode and frequency combination.

Horizontal jitter at high dot clocks.

> This problem shows especially when drawing operations such as scrolling are in progress. If you're using a 542x/3x/46/6x/754x, try the `"fifo_conservative"` option. Failing that, you can try the `"fast_dram"` option, or use a lower dot clock. If that is not sufficient, the `"noaccel"` option or `"no_bitblt"` will probably help. When using a 546x, option `"fifo_aggressive"` can also be tried.

`Wavy' screen.

> Horizontal waving or jittering of the whole screen, continuously (independent from drawing operations). You are probably using a dot clock that is too high; it is also possible that there is interference with a close MCLK. Try a lower dot clock. You can also try to tweak the mode timings; try increasing the second horizontal value somewhat. Here's a 65 MHz dot clock 1024x768 mode (about 60 Hz) that might help:
>
> ```
> "1024x768"     65      1024 1116 1228 1328     768  783  789  818
> ```
>
> If you are using programmable clocks with Clockchip `"cirrus"`, try disabling it and using the default set of clocks.

Crash or hang after start-up (probably with a black screen).

> Try the `"noaccel"` option. If that works, try Option `"no_bitblt"` for somewhat better performance. Check that the BIOS settings are OK; in particular, disable caching of 0xa0000-0xaffff. Disabling hidden DRAM refresh may also help.

Crash, hang, or trash on the screen after a graphics operation.

This may be related to a bug in one of the accelerated functions, or a problem with the BitBLT engine. Try the `"noaccel"` option, or the `"no_bitblt"` option. Also check the BIOS settings.

`Blitter timeout' messages from the server.

Same as for the above entry.

Screen is `wrapped' vertically. (542x/3x/46)

This indicates a DRAM configuration problem. If your card has two megabytes of memory, try the `"no_2mb_banksel"` option, or use `videoram "1024"` if you only use 1 Mbyte for the virtual screen.

Corrupted text in terminal window.

This has been reported on non-standard video implementations. Use the `"no_bitblt"` option.

Streaks or hangs with laptop chipset

This can happen if the dot clock is high enough to leave very little bandwidth for drawing (e.g. 40 MHz on a 512K card), and (5422-style) acceleration is used.

Occasional erroneous pixels in text, pixel dust when moving window-frame

Probably related to MCLK setting that is too high (can happen with linear addressing even though banked mode runs OK).

Chipset is not detected.

Try forcing the chipset to a type that is most similar to what you have.

Incorrect little lines (mostly white) appear occasionally

This may be related to a problem with system-to-video-memory BitBLT operations. Try the `"no_imageblt"` option if it annoys you.

Textmode is not properly restored

This has been reported on some configurations. In XFree86 3.1 the SVGA server probe would corrupt a register on the 543x, requiring a Chipset line. Normally you should be able to restore the textmode font using a utility that sets it (`setfont`, `runx`, `restorefont` on Linux).

Erratic system behaviour at very high dot clocks

It is possible that high dot clocks on the video card interfere with other components in the system (e.g. disk I/O), because of a bad card and/or motherboard design. It has been observed on some PCI 5428-based cards (which are very rare, since the 5428 chip doesn't support PCI).

No mouse cursor, or cursor appears twice on screen

With high dot clocks, the graphics card's hardware cursor doesn't operate correctly. Try option `"sw_cursor"` or use a lower screen refresh.

Random/garbage pixels on far right or bottom of screen (546x)

This problem is usually associated with using a virtual screen size larger than the screen display size. The garbage pixels are unused portions of the frame buffer that result from padding each scanline to an integral number of memory tiles. To eliminate the extra pixels, use a screen display

mode whose pixel width is evenly divisible by 128 / bits per pixel.

Screen is wrapped horizontally on right side (546x)

Same as above entry.

The screen is initially displayed correctly, but then turns all white. (546x)

This problem usually happens at high bit depths and while the screen is changing rapidly (catting a long file or dragging a large window around). The RamBus memory is being overdriven. Use Option `"med_dram"`, or, if the problem persists, Option `"slow_dram"`.

For other screen drawing related problems, try the `"noaccel"` option (if `"no_bitblt"` doesn't help).

If are having driver-related problems that are not addressed by this document, or if you have found bugs in accelerated functions, you can try contacting the XFree86 team (the current driver maintainer, Corin Anderson, can be reached at *corina@the4cs.com*), or post in the Usenet newsgroup "*comp.windows.x.i386unix*".

In fact, reports (success or failure) are very welcome, especially on configurations that have not been tested. You can do this via the BetaReport form (mail to report@XFree86.org). You may want to keep an eye on forthcoming beta releases at *www.xfree86.org*.

Information for Cirrus Chipset Users **:** *Trouble shooting with the ``cirrus'' driver*
Previous: *The ``cl64xx'' Driver*
Next: *Tested Configurations*

8. Tested Configurations

Version 3.3.3 has had the following configurations tested:

CL-GD5446 with 2MB memory on PCI bus

CL-GD5464 with 2MB memory on PCI bus

CL-GD5465 with 4MB memory on PCI bus

CL-GD5480 with 4MB memory on PCI bus

CL-GD5465 with 4MB memory on AGP bus

For version 3.3, the following configurations have received a certain amount of testing:

CL-GD5446 with 2MB memory on PCI bus

> Support for dot clocks > 85 MHz has been fixed. At 16bpp, it has been reported that some stippled edges of window frames may be corrupted or show the wrong colors. The option "xaa_no_pixmap_cache" eliminates the problem.

CL-GD5464 with 4MB memory on PCI bus

CL-GD7543 on PCI bus

This is a list of configurations that has received testing with one or more of the changes introduced in version XFree86 3.2A. The amount of testing is very small for some of the configurations, and the summaries may be incomplete. If you can contribute, please do so. For the latest information check the latest version of this document on *www.xfree86.org*.

CL-GD5426 on VL-bus

> This configuration was only tested with an early version of the XAA code.

CL-GD5434 with 2MB memory on VL-bus

> MMIO operation is supported. This configuration was only tested with an early version of the XAA code.

CL-GD5436 with 2MB memory on PCI-bus

> Works OK. Non-MMIO operation might have problems.

CL-GD5446 with 2MB memory on PCI bus

> Works OK in MMIO mode. 32bpp probably doesn't work. The support for dot clocks > 85 MHz at 8bpp may or may not work.

CL-GD5462 with 2MB memory on PCI bus

CL-GD5462 with 4MB memory on PCI bus

CL-GD5464 with 4MB memory on PCI bus

TOC

Works OK at 8bpp, 16bpp, 24bpp and 32bpp. CL-GD5464 works OK at 16bpp, -weight 555.

CL-GD7543 on PCI bus

Works for 8bpp, 16bpp on TFT display (TI TravelMate 5000). Although the previous version, 3.2, was reported to broken, on some configurations it worked, while others were reported not to work correctly. On 800x600 displays, the recommended dot clock is 40 MHz for TFT and 33.7 MHz for a DSTN panel, with corresponding horizontal syncs of 33.7 kHz for TFT and 38.6 kHz for DSTN. However, reports indicate that the VESA standard 40 MHz 800x600 timing may cause problems. The solution is decrease the fourth horizontal timing number or use a dot clock of 36 MHz.

Some configurations for which no up-to-date testing data is available:

CL-GD5429 on VL-bus

BitBLT operation should be fixed in 3.2. MMIO does not work, but not tested with with 3.2 or 3.2A.

CL-GD5430 on PCI-bus

Works OK. 24bpp was broken, but should be fixed in later versions (3.2A).

CL-GD5430, and CL-GD5436 and CL-GD5446 with 1MB memory

It would be nice to know whether these chips needs the same treatment at 16bpp as the CL-GD5434 with 1MB memory does.

CL-GD5434 with 1MB memory on PCI bus

8bpp, 16pp and 24bpp work OK. 16bpp no longer has "static" problems. MMIO operation is supported.

CL-GD5436 and CL-GD5446 with 1MB memory

In particular the FIFO settings for this configuration are uncertain.

CL-GD7541

CL-GD7548

Should be compatible with 7543, but untested. Reports indicate that it worked with 3.2, and there's no reason why it shouldn't work with 3.2A.

Information for Cirrus Chipset Users **:** *Tested Configurations*
Previous: *Trouble shooting with the ``cirrus'' driver*
Next: *Driver Changes*

9. Driver Changes

Changes since XFree86 3.3.2:

- Fix transparent screen-to-screen copies on 546x.
- The built-in screen saver now correctly blanks the screen on 546x chips.
- Driver prevents the use of the HW cursor on the 546x when the screen height is greater than 1023 scanlines (fix to double pointer problem).
- CPU-to-screen BitBLT transfers disabled on the 5465. This fix should prevent 5465 AGP lockups.
- Fixed mode display problem with 5480 at high resolutions.

Changes since XFree86 3.2A:

- A bug that caused a server crash with memory-mapped I/O operation on some chips has been fixed.
- Correct handling of dot clocks > 85 MHz on the 5436 and 5446.
- Preliminary support for the CL-GD7555 (no detection yet).
- Support has been added for the CL-GD5480 and CL-GD5465.
- 32bpp mode has been fixed on some Alpine family chips.
- Support for dot clocks up to 230 MHz has been added for Laguna family chips.

Changes since XFree86 3.2:

- Enhanced acceleration using XAA on all chips with a BitBLT engine.
- Enhanced acceleration using XAA for the Laguna series (546x).
- 24bpp mode on 5430 is fixed.
- Improved support for 754x, including support for LCD stretching/centering.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/cirrus.sgml,v 3.23.2.6 1998/11/07
13:37:51 dawes Exp $
```

```
$XConsortium: cirrus.sgml /main/12 1996/10/28 05:43:32 kaleb $
```

Information for Trident Chipset Users

TOC

The XFree86 Project, Inc.

June 25 1999

1. Supported chipsets

2. Special considerations for 512k boards

3. Additional Notes

1. Supported chipsets

The Trident driver has undergone some slight work for XFree86 3.3.3. Because of this work, all of the Trident SVGA chipsets, except the very first one, are supported by both the color and monochrome servers.

8800CS 8200LX 8900B 8900C 8900CL/D 9000 9000i 9100B 9200CXr 9320LCD 9400CXi 9420 9420DGi 9430DGi 9440AGi 9660XGi 9680 ProVidia9682 ProVidia9685 Cyber9382 Cyber9385 Cyber9385-1 Cyber9388 Cyber9397 Cyber9520 Cyber9525 3DImage975(PCI) 3DImage975(AGP) 3DImage985(AGP) Blade3D CyberBlade

It must be noted that the 9000i chipset is treated as a 9000 by the server. Additionally the 9100B is treated as a Trident 8900CL. Therefore it is equivalent to putting `` `Chipset "tvga8900cl"' `` or `` `Chipset "tvga9000"' `` in the `XF86Config` file. Also, note that the 9000i, 9100B have not been tested with the server, but should work in this way according to the chipset documentation.

NOTES:

- The chipset keyword changed in XFree86 v3.3.2 and now you no longer specify 'tgui96xx' as the generic keyword, but you actually specify your chip. i.e. Chipset 'tgui9685' will set a ProVidia9685 chip.
- The Cyber9388/9397, 3DImage975 and 3DImage985 cards are fixed in XFree86 v3.3.3, these chipsets have some acceleration now too. This acceleration has been disabled by default for the Cyber9388/9397 because there have been problems, but it can be re-enabled with the `"accel"` option (see below).
- 24bpp is all drivers remains unaccelerated, this will change in a future version, although 32bpp acceleration is supported for all TGUI based chipset except the 9440 which doesn't have the capability.
- 16bpp is now supported for the Cyber9320 chipset.

Option "nolinear"

Turn off linear mapping

Option "linear"

Force linear mapping. Use this if you have a non-PCI card and require 16bpp support. Note: ISA cards can only access up to 16MB of memory, so be sure you have less than this or it could cause a system hang.

MemBase 0x???????

This option may be used to specify the start address of the linear frame buffer. By default for VLBus/EISA cards it is at 60MB. For the 8900CL/D, it is at 15MB.

Option "no_mmio"


TOC

This option turns off Memory Mapped IO support. MMIO is enabled by default when acceleration is enabled. Acceleration doesn't work well when MMIO is disabled.

Option "tgui_pci_read_on"

Turn on PCI burst read mode.

Option "tgui_pci_write_on"

Turn on PCI burst write mode.

Option "pci_burst_on"

Turn on PCI burst (read and write)

Option "pci_burst_off"

Turn off PCI burst (read and write)

NOTE: PCI burst modes are now OFF by default for TGUI9440 cards because it often upsets its Graphics Accelerator. It can be turned it back on as may improve performance. PCI burst modes are ON by default for all other PCI/AGP cards.

ClockChip "tgui"

Turn on programmable clocks. This is the default for TGUIs.

Option "no_program_clocks"

Turn off programmable clock. Use fixed VGA clocks only. Useful for fixed frequency monitors - usually used for VGA monitors - not SVGA.

Option "noaccel"

Turn off XAA acceleration.

Option "accel"

Enable acceleration for the Cyber9388/9397.

Option "xaa_no_color_exp"

Disable color expansion.

Option "no_stretch"

Disable LCD stretching on Cyber 938x based chips.

Option "lcd_center"

Enable LCD centering on Cyber 938x based chips.

Option "cyber_shadow"

Enable Shadow registers, might be needed for some Cyber chipsets. (laptop machines)

Option "tgui_mclk_66"

Pushes the Memory Clock from its default value to 66MHz. Increases graphics speed dramatically, but use entirely at your own risk, as it may damage the video card. If snow

appears, disable. Only tested on the 9440.

The original Trident chipset, 8800BR, cannot be supported as an SVGA chipset by either the color or monochrome servers. The chip is supported, however, by the ``generic'' driver for the monochrome server.

Information for Trident Chipset Users **:** *Supported chipsets*
Previous: *Information for Trident Chipset Users*
Next: *Special considerations for 512k boards*

2. Special considerations for 512k boards

There are no longer any special considerations for 512k Trident boards. The driver is now configured so that they can use modes with normal timings. The available pixel clocks are halved compared with those specified on the Clocks line

Be aware that older Trident chipsets support a maximum clock of 65Mhz. Hence the best actual clock available to the color server is 32.5Mhz. This means, in broad terms, that the color server will require an interlaced mode to be defined for resolutions above 640x480. Newer chipsets (8900CL, 9000, 9000i, 9100B, 9200CX and 9420) support up to 16 clocks, and can support much higher clocks, which will allow 800x600 modes, non-interlaced.

3. Additional Notes

We have had reports of the server failing to detect the amount of installed memory and the correct dot-clocks on older TVGA8900 boards. If the server fails to detect the correct amount of memory, use the `"Videoram"` keyword in your `XF86Config` file to specify it. (e.g. `Videoram 512` or `Videoram 1024`). If the server has problems detecting the dot-clocks, try adding the following line to your `XF86Config` file:

```
Clocks  25 28 45 36 57 65 50 40
```

This line gives the clock values provided by older Trident clock synthesizer chipsets. This also appears to be the standard first 8 clocks for the newer clock synthesizers, but you should have no problems on newer boards.

Some newer Trident 8900B/C boards are apparently being built with the clock synthesizers used on the 9000 and 8900CL boards. If your board has a chip labeled "Trident TCK900x" ("x" has been seen as 2 or 4; there may be others), your board may actually have a 4th clock select bit. The 9002 has twelve distinct clocks (the other 4 are duplicates); the 9004 has 16 clocks (the same 12 as the 9002 + 4 others). If you see such a chip on a board with an 8900B or 8900C, put the following line in the Device section of your `XF86Config` file:

```
Option "16clocks"
```

This will cause the same clock selection code as is used for the 8900CL to be used for the board.

While developing the Trident driver, an interesting and perturbing hardware phenomenon was discovered. When using the default board jumper configuration, dot-clocks above 57Mhz would frequently lock up the machine. There appear to be jumpers on all of the Trident boards that determine whether the board will operate in zero-wait-state mode on the ISA bus. Disabling the zero-wait-state mode via jumpers cured the lockups, but at the expense of performance. Whether or not a given system will experience this problem is likely a combination of (a) bus speed, (b) video memory speed, and (c) dot clock speed. So be prepared for this phenomenon to occur, and have the board documentation handy.

NOTE: VLBus cards are also subject to the above. By specifying the Clocks in the `XF86Config` file, these lockups are overcome. But it may be worth checking wait states etc. on the card and in the BIOS setup.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/trident.sgml,v 3.22.2.9 1999/06/25
08:57:15 hohndel Exp $
```

```
$XConsortium: trident.sgml /main/11 1996/10/28 04:24:08 kaleb $
```

Information for NeoMagic Users NeoMagic Driver Version 2.0.0

The XFree86 Project Inc.

4 November 1998

1. Supported hardware
2. Features
3. Technical Notes
4. Reported Working Laptops
5. Configuration
6. Driver Options
7. Known Limitations
8. Authors

1. Supported hardware

- NeoMagic 2200 (MagicMedia256AV)
- NeoMagic 2160 (MagicGraph128XD)
- NeoMagic 2097 (MagicGraph128ZV+)
- NeoMagic 2093 (MagicGraph128ZV)
- NeoMagic 2090 (MagicGraph128V)
- NeoMagic 2070 (MagicGraph128)

2. Features

- Full support for internal flat panels, external monitors, and simultaneous internal/external displays.
- Complete set of Panel Resolutions supported including stretch and centering modes for running lower resolutions on fixed resolution panels.
- Support for depths of 8, 15, 16 and 24 bits per pixel.
- Hardware Cursor support to reduce sprite flicker.
- Hardware accelerated drawing engine for 8, 15 and 16 bit per pixel modes.
- Fully programmable clocks supported in external monitor only mode.
- Robust text mode restore for VT switching.

TOC

3. Technical Notes

- Enable both internal "intern_disp" and external "extern_disp" options to get simultaneous panel/CRT support.

4. Reported Working Laptops

- Acer Travelmate 7120T
- Acer Extensa 367, 367D & 710TE
- Actebis TN559Pro
- Asus P6300
- CTX EzBook 700 & 77X series
- Compaq Presario 1080, 1210, 1215, 1220, 1610, 1611, 1620, 1621 & 1640
- Dell Inspiron 3000 & 3200
- Dell Latitude CP, CPi, LM & XPi
- Digital VP HiNote 575, 703, 717 & 720
- FIC DESIGNote 5550
- Fujitsu LifeBook 420D & 656Tx
- Gateway 2000 Solo 2300XL, 2500LS & 5150
- Highscreen XD Advance II 21,1" TFT
- Hi-Grade Notino AS6000 pII/266Mhz
- Hitachi VisionBook Plus 5000
- HP Omnibook 800, 3000, 3100, 4100 & Sojourn
- IBM ThinkPad 380D, 380E, 380ED, 380XD, 385XD, 560X & 600
- LEO DESIGNote 5550
- Micron Transport XKE
- NEC Ready 330T
- NEC Versa 2780 MT, 5060X, 5080X, 6060 & 6230
- NEC MB12C/UV (mobio NX)
- OPTI Phoenix
- Panasonic CF_S21, CF-25 MKIII & CF-35
- Quantex H-1330
- Sceptre 4500
- SEH DESIGNote 5550
- Siemens Nixdorf Scenic 510
- Sony PCG-505, PCG-705, PCG-717, PCG-719 & PCG-731
- TI Extensa 660 CDT


TOC

- Toshiba Libretto 100CT
- Toshiba Protege SS3000
- UMAX 520T

5. Configuration

The driver auto-detects all device info included memory size, so use the following device section in your XF86Config file:

```
        Section "Device"
                Identifier      "NeoMagic"
        EndSection
```

or let xf86config or XF86Setup do this for you.

But if you have problems with auto-detection, you can specify:

```
      VideoRam  - in kilobytes
      DacSpeed  - in MHz
      MemBase   - physical address of the linear framebuffer
      MMIOBase  - physical address of the memory mapped IO registers
```

6. Driver Options

- "linear" - linear framebuffer mode (default)
- "no_linear" - banked framebuffer mode
- "no_accel" - software rendering only
- "hw_cursor" - hardware cursor requested (default)
- "sw_cursor" - software cursor only
- "mmio" - use I/O space via memory map (default)
- "no_mmio" - use I/O space directly
- "intern_disp" - enable internal display (default)
- "extern_disp" - enable external display
- "no_stretch" - disable stretching of lower resolution modes on panel
- "lcd_center" - center lower resolution modes on panel

NOTE: Stretching of panel image is on by default for lower panel resolutions.

Options useful for special lcd mode setting (should not be needed):

- "prog_lcd_mode_regs" - set special lcd mode registers (2070 default)
- "no_prog_lcd_mode_regs" - don't set lcd mode registers (non-2070 default)
- "prog_lcd_mode_stretch" - force lcd mode regs if stretching is enabled
- "no_prog_lcd_mode_stretch" - no lcd mode regs if stretching (default)

Option for subnotebooks and other laptops with uncommon size panels:

- "override_validate_mode" - disable LCD mode checking

WARNING: Disabling mode checking will allow for invalid modes that could damage your LCD.

7. Known Limitations

- External monitor support on the NM2070.
- Banked, or no_linear mode on the NM2070.
- Horizontal centering for lower than panel resolution on NM2070.

8. Authors

Jens Owen *jens@precisioninsight.com* Kevin E. Martin *kevin@precisioninsight.com*

This driver was donated to The XFree86 Project by Precision Insight, Inc. Cedar Park, TX USA

http://www.precisioninsight.com

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/neo.sgml,v 1.1.2.2 1998/11/13
13:00:46 dawes Exp $
```

Information for Rendition Users

TOC

The XFree86 Project Inc.

1 August 1999

1. Supported hardware
2. Important notices
3. Features
4. XF86Config Option
5. News in this release
6. Major fixes in this release
7. Known problems in current driver
8. Work in progress (not finished in time for release)

1. Supported hardware

All cards based on the V1000 or the V2x00 should be supported. The server was tested on a miroCRYSTAL VRX (V1000), Intergraph Intense-100 3D (V1000), Diamond Stealth II S220 (V2100), Hercules Thriller3D (V2200) and Innovision Warrior3D (V2200).

2. Important notices

V1000 cards can only work as primary display card due to hardware limitations.

Some V1000-based videocards are known to lock up the computer if you have write-combine activated. Disabling it removes the problem. Look for settings in the motherboards BIOS and disable ALL settings that has to do with write-combine (usualy called USWC or just WC for short).

The "chipset" option is now implemented and honored when used. Unfortunatly some legacy-code in the driver is preventing it from working with any cards but the primary display card.

If you have problems with hardware cursor use the "sw_cursor" option to revert back to software cursor.

3. Features

- Unaccelerated
- Hardware cursor
- Supported color depths
  - 8 bits per pixel (256 pseudo colour)
  - 15 bits per pixel (actualy 16-bits with RGB-weight 555, 32768 colors)
  - 16 bits per pixel (high colour, RGB-weight 565, 65536 colors)
  - 32 bits per pixel (true colour, sparse 24bit, 16M colors)

TOC

4. XF86Config Option

Option "sw_cursor"

Disables use of the hardware cursor.

Option "overclock_mem"

Run the memory at a higher clock. Useful on some cards with display glitches at higher resolutions. But adds the risk to damage the hardware. Use with caution.

DacSpeed "MHz"

Set custom ramdac limit. We have currently no way of knowing if the v2x00 chip is a v2100 (170MHz) or v2200 (203MHz and 230MHz) so we assume the lowest. Use this option to manualy override the value.

5. News in this release

- Hardware-cursor on V2x00 cards.

6. Major fixes in this release

- Depth 15 works on V1000 cards.
- Bandwith limits are now included. Can be overrided with DacSpeed option.

7. Known problems in current driver

- Displays with depth 15 ("-bpp 15" or "-bpp 16 -weight 555") has problems on V2x00 cards.
- Switching from display to VC and back to display can lock up V2x00 cards.
- When scrolling the virtual display on a V1000 card parts of the screen will become distorted. Problem dissapears when you continue moving around. V2x00 does not exhibit this problem. Probably a bug in the driver rather than a limitation of the chip.
- Option "chipset" is honored. Unfortunatly the driver still has problems and will only work if the rendition card is the primary display card in the system.
- Switching to VC does not restore correct textmode. Instead it defaults to 80x25.
- A horizontal distortion around the hardware cursor can be seen on certain modes. It can be fixed by turning off hardware cursor or by lowering the required bandwith of the mode.

8. Work in progress (not finished in time for release)

- Acceleration for V1000 chipset. Some bugs to clear out.
- Acceleration for V2100 and V2200 chipset. Not ready for general use.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/rendition.sgml,v 1.1.2.10
1999/08/17 07:39:31 hohndel Exp $
```

Information for EPSON SPC8110 Users

Thomas Mueller (tmueller@sysgo.de)

October 15, 1998

1. General Notes

2. `XF86Config` options

3. Video modes

    3.1. Clocks

    3.2. Example Modes

4. Acknowledgments

1. General Notes

This server provides support for the Seiko/EPSON SPC8110F0A LCD VGA controller chip.

The driver was developed and tested using an EPSON 486D4 CardPC using CRT display mode. LCD operation was successfully tested using an earlier release of this driver.

The current driver has support for linear mapping of the frame buffer, supports the hardware cursor and uses the Bitblt engine for basic operations such as CopyArea and solid fills.

2. XF86Config options

The driver should be able to probe the presence of a SPC8110 chip. If the driver fails to probe the chip correctly define the chip explicitly in the screen section.

Device Section Entries and Options Currently Supported:

Chipset "spc8110"

May be specified if probing fails or to accelerate server startup. The value must be "spc8110".

VideoRam kilobytes

If specified the value (in kilobytes) will be used, otherwise the amount of memory will be probed on startup.

Option "sw_cursor"

Disables the use of the hardware cursor. The hardware cursor requires one Kbyte of video memory as pattern storage area. If you need the full amount of video memory you may want to disable the hardware cursor using this option. Also the hardware cursor code was not tested with cursor images larger than 64 pixels (high or wide), so if you use large images you may have to disable the hardware cursor.

Option "no_linear"

Disables the use of the linear aperture. If this option is set the driver will use the standard VGA memory window at 0xa0000 otherwise it will map the whole video memory.

Membase baseaddress

In VLB/486LB configuration the linear aperture address will be set to 0x03E0.0000, in PCI configuration the address will be read from the PCI configuration space. The base address in VLB/486LB systems may be set to any value using the "Membase" definition.

Option "noaccel"

Disables the use of the Bitblt engine. Normally the driver accelerates screen-to-screen copy operations and solid fills.

Since the SPC8110 puts certain restrictions on the use of the Bitblt engine you will notice different performance between certain operations (eg window movement). If this is a problem for your application you may want to disable the accelerator.

Option "fifo_moderate"

Option "fifo_conservative"

Usually the driver computes the FIFO threshold values for the SPC8110's write buffer correctly.

However for certain modes (eg the 832x624 mode shown below) the FIFO is programmed too aggressively which leads to streaks in the display during screen updates. With option "fifo_moderate" the computed FIFO low request level is incremented by one with "fifo_conservative" it is incremented by two.

---

3. Video modes

The driver probes wether the chip is configured for CRT only or LCD/simultaneous mode of operation. In the former case it will enable clock programming and will support any mode which is within the limits of the hardware. If the chip is configured for LCD/simultaneous operation mode the driver will respect the settings of the BIOS and allow only one video mode conforming with the panel size.

The driver does not support interlaced or double scan modes.

3.1. Clocks

Probing is supported, but of course the usual warnings and disclaimers apply. Probing may momentarily subject your monitor/panel to sweep frequencies in excess of its rating. The cautious may wish to turn off the monitor while the probe is running. In CRT mode the driver may produce video timings inadequate for your monitor, handle with care!

As with many integrated designs the speed of the graphics operations depend very much of the refresh rate you use. Higher refresh rates yield lower performance.

3.2. Example Modes

The following XF86Config "Device" section should work for all configurations:

```
Section "Device"
    Identifier "CardPC"
    VendorName "EPSON"
    BoardName  "CardPC"
    Chipset    "spc8110"
    Option    "sw_cursor"
    Membase   0x03e00000
    Option    "no_linear"
    Option    "noaccel"
    Option    "fifo_moderate"
EndSection
```

This Modeline was tested in a 640x480 panel configuration:

```
Modeline "640x480" 28.36   640 672 768 800  480 490 492 525
```

These Modelines were tested in a CRT configuration:

```
  Modeline "640x480" 25.175  640 664  760  800  480 491 493 525
  Modeline "800x600" 36      800 824  896 1024  600 601 603 625
  ModeLine "832x624" 40      832 873 1001 1090  624 625 627 651
  Modeline "640x400" 25.175  640 664  760  800  400 409 411 450
```

TOC

Information for EPSON SPC8110 Users : *Video modes*

Previous: *`XF86Config` options*

Next: *Acknowledgments*

4. Acknowledgments

Thanks to Epson Europe Electronics and ProBIT GmbH, Berlin for providing a loaner system and documentation to get things started.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/epson.sgml,v 1.1.2.3 1998/11/13
13:00:46 dawes Exp $
```

Information for 3DLabs Chipset Users

The XFree86 Project Inc.

25 Juni 1999

1. Supported hardware

2. Features

3. XF86Config Option

4. Bugs and Limitations

5. Authors

TOC

1. Supported hardware

This server supports the following 3DLabs chipsets:

- GLINT 500TX with IBM RGB526 RAMDAC
- GLINT MX plus Delta with IBM RGB526 and IBM RGB640 RAMDAC
- GLINT MX plus Gamma with IBM RGB526 and IBM RGB640 RAMDAC
- Permedia with IBM RGB526 RAMDAC
- Permedia 2 (classic, 2a, 2v)

TOC

2. Features

- accelerated
- hardware cursor
- DPMS support
- supported color depths
  - GLINT MX/500TX: 8/16/32 bpp
  - Permedia: 8/16/32 bpp
  - Permedia 2: 8/16/24/32 bpp

Information for 3DLabs Chipset Users : *Features*
Previous: *Supported hardware*
Next: *XF86Config Option*

3. XF86Config Option

Option "sw_cursor"

disable the hardware cursor.

Option "no_pixmap_cache"

disables use of the pixmap cache. Might be useful if drawing errors occur.

Option "no_accel"

completely disables acceleration. Usually not recommended.

Option "pci_retry"

stall the PCI bus while the graphics engine is busy. While this might give slightly higher performance, you run the risk of disturbing other devices that are waiting to be serviced by the processor. This option may cause problems with SCSI cards, serial connections, sound cards, etc.

Option "firegl_3000"

needed for the Diamond Fire GL 3000 in order to use the primary output on that card. The second screen is currently not supported.

Option "overclock_mem"

Run the memory at a higher clock. Useful on some cards with display glitches at higher resolutions. But adds the risk to damage the hardware. Use with caution.

Option "sync_on_green"

Composite sync on green. Possibly useful if you wish to use an old workstation monitor. This feature is only implemented for Permedia 2 based boards (Permedia 2v doesn't have this capability). Default is to use positive sync polarity. As many SOG monitor want negative sync polarity, you'll have to play around with the "-HSync" and "-VSync" Modeline flags if you own such a monitor.

4. Bugs and Limitations

- The 500TX and MX chipsets cannot switch modes, therefore only the first mode on the modes line is available.
- In some color depths without acceleration there are color problems.
- While the server is accelerated, there is room for improvement. As our development is focusing on XFree86-4.0 we are not planning to change that in the 3.3.x branch. XFree86-4.0 will include a significantly faster server.

5. Authors

- Alan Hourihane <*alanh@fairlite.demon.co.uk*>
- Dirk Hohndel <*hohndel@XFree86.org*>
- Stefan Dirsch <*sndirsch@suse.de*>
- Helmut Fahrion <*hf@suse.de*>
- Special thanks to Elsa AG, Aachen for making it possible for us to develop this server and furnishing us with plenty of boards and information to help us on the way
- Very special thanks to SuSE GmbH, Nuernberg for allowing some of us to work on this server on paid time, thereby financing development of this server.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/3DLabs.sgml,v 1.1.2.5 1999/08/17
07:39:28 hohndel Exp $
```

Information for i740 Users

TOC

Precision Insight, Inc.

18 February 1999

1. Supported Hardware
2. Features
3. Technical Notes
4. Reported Working Video Cards
5. Configuration
6. Driver Options
7. Known Limitations
8. Author

TOC

1. Supported Hardware

- Intel 740 based cards

2. Features

- Full support for 8, 15, 16, 24 and 32 bit per pixel depths.
- Hardware cursor support to reduce sprite flicker.
- Hardware accelerated 2D drawing engine support for 8, 15, 16 and 24 bit per pixel depths.
- Support for high resolution video modes up to 1600x1200.
- Support for doublescan video modes (e.g., 320x200 and 320x240).
- Support for gamma correction at all pixel depths.
- Fully programmable clock supported.
- Robust text mode restore for VT switching.

3. Technical Notes

- Hardware acceleration is not possible in 32 bit per pixel depth.
- Interlace modes cannot be supported.

4. Reported Working Video Cards

- Real3D Starfighter AGP
- Real3D Starfighter PCI
- Diamond Stealth II/G460 AGP
- 3DVision-i740 AGP
- ABIT G740 8MB SDRAM
- Acorp AGP i740
- AGP 2D/3D V. 1N, AGP-740D
- AOpen AGP 2X 3D Navigator PA740
- ARISTO i740 AGP (ART-i740-G)
- ASUS AGP-V2740
- Atrend (Speedy) 3DIO740 AGP (ATC-2740)
- Chaintech AGP-740D
- EliteGroup(ECS) 3DVision-i740 AGP
- EONtronics Picasso 740
- EONtronics Van Gogh
- Everex MVGA i740/AG
- Flagpoint Shocker i740 8MB
- Gainward CardExpert 740 8MB
- Genoa Systems Phantom 740
- Gigabyte Predator i740 8MB AGP
- Hercules Terminator 128 2X/i AGP
- HOT-158 (Shuttle)
- Intel Express 3D AGP
- Jaton Video-740 AGP 3D
- Jetway J-740-3D 8MB AGP, i740 AGP 3D
- Joymedia Apollo 7400
- Leadtek Winfast S900
- Machspeed Raptor i740 AGP 4600
- Magic-Pro MP-740DVD
- MAXI Gamer AGP 8 MB


TOC

- Palit Daytona AGP740
- PowerColor C740 (SG/SD) AGP
- QDI Amazing I
- Soyo AGP (SY-740 AGP)
- Spacewalker Hot-158
- VideoExcel AGP 740
- ViewTop ZeusL 8MB
- Winfast S900 i740 AGP 8MB

Information for i740 Users **:** *Reported Working Video Cards*
Previous: *Technical Notes*
Next: *Configuration*

5. Configuration

The driver auto-detects all device information necessary to initialize the card. The only lines you need in the "Device" section of your XF86Config file are:

```
Section "Device"
    Identifier "i740"
EndSection
```

or let xf86config or XF86Setup do this for you.

However, if you have problems with auto-detection, you can specify:

- VideoRam - in kilobytes
- DacSpeed - in MHz
- MemBase - physical address of the linear framebuffer
- IOBase - physical address of the memory mapped IO registers

6. Driver Options

- "hw_cursor" - request hardware cursor (default)
- "sw_cursor" - software cursor only
- "no_accel" - software rendering only
- "sgram" - force the use of SGRAM timing info
- "sdram" - force the use of SDRAM timing info

Note: the i740 X server should automatically detect whether your card has SGRAM or SDRAM. Use the "sgram" and "sdram" options if it is incorrectly detected.

7. Known Limitations

- Certain drawing operations are very slow when using 24 bit per pixel depth mode. We hope to fix this in a future release.

8. Author

- Kevin E. Martin <*kevin@precisioninsight.com*>

This driver was donated to The XFree86 Project by:

```
    Precision Insight, Inc.
    Cedar Park, TX
    USA
```

http://www.precisioninsight.com

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/i740.sgml,v 1.1.2.1 1999/04/15
12:04:31 hohndel Exp $
```

TOC

Copyright

The XFree86 Project, Inc.

1. XFree86 Copyright

2. Other Copyrights

2.1. X Consortium

2.2. Berkeley-based copyrights:

2.3. NVidia Corp

1. XFree86 Copyright

XFree86 code without an explicit copyright is covered by the following copyright:

2. Other Copyrights

Portions of code are covered by the following copyrights:

2.1. X Consortium

2.2. Berkeley-based copyrights:

2.2.1. General

2.2.2. UCB/LBL

This software was developed by the Computer Systems Engineering group at Lawrence Berkeley Laboratory under DARPA contract BG 91-66 and contributed to Berkeley.

All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Lawrence Berkeley Laboratory.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2.3. NVidia Corp

NOTICE TO USER: The source code is copyrighted under U.S. and international laws. NVIDIA, Corp. of Sunnyvale, California owns the copyright and as design patents pending on the design and interface of the NV chips. Users and possessors of this source code are hereby granted a nonexclusive, royalty-free copyright and design patent license to use this code in individual and commercial software.

Any use of this source code must include, in the user documentation and internal comments to the code, notices to the end user as follows:

NVIDIA, CORP. MAKES NO REPRESENTATION ABOUT THE SUITABILITY OF THIS SOURCE CODE FOR ANY PURPOSE. IT IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND. NVIDIA, CORP. DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOURCE CODE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA, CORP. BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOURCE CODE.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/CPYRIGHT.sgml,v 3.9.2.3 1999/06/25
08:57:13 hohndel Exp $
```

TOC

```
$XConsortium: CPYRIGHT.sgml /main/5 1996/10/25 16:24:53 kaleb $
```

Copyright **:** *Other Copyrights*
Previous: *XFree86 Copyright*
Next: *Copyright*

Mouse Support in XFree86

Kazutaka Yokota

20 April 1999

1. Introduction

2. Supported Hardware

3. OS Support for Mice

3.1. Summary of Supported Mouse Protocol Types
3.2. BSD/OS
3.3. FreeBSD
3.4. FreeBSD(98)
3.5. Interactive Unix
3.6. Linux
3.7. Linux/98
3.8. LynxOS
3.9. NetBSD
3.10. NetBSD/pc98
3.11. OpenBSD
3.12. OS/2
3.13. SCO
3.14. Solaris
3.15. SVR4
3.16. PANIX

4. Configuring Your Mouse

5. XF86Config Options

5.1. Buttons
5.2. ZAxisMappping
5.3. Resolution

TOC

6. Mouse Gallery

6.1. MS IntelliMouse (serial, PS/2)

6.2. Kensington Thinking Mouse (serial, PS/2)

6.3. Genius NetScroll (PS/2)

6.4. Genius NetMouse and NetMouse Pro (serial, PS/2)

6.5. ALPS GlidePoint (serial, PS/2)

6.6. ASCII MieMouse (serial, PS/2)

6.7. Logitech MouseMan+ and FirstMouse+ (serial, PS/2)

1. Introduction

This document describes mouse support in XFree86 3.3.4, whose X servers have the revised mouse driver.

Mouse configuration has often been mysterious task for novice users. However, once you learn several basics, it is straightforward to choose options in `XF86Setup` or write the `"Pointer"` section in the `XF86Config` file by hand.

2. Supported Hardware

XFree86 X servers support three classes of mice: serial, bus and PS/2 mice.

Serial mouse

The serial mouse has been the most popular pointing device for PCs. There have been numerous serial mouse models from a number of manufactures. Despite the wide range of variations, there have been relatively few protocols (data format) with which the serial mouse talks to the host computer.

The modern serial mouse conforms to the PnP COM device specification so that the host computer can automatically detect the mouse and load an appropriate driver. The XFree86 3.3.2 X servers support this specification and can detect popular PnP serial mouse models.

Bus mouse

The bus mouse connects to a dedicated interface card in an expansion slot. Some video cards, notably those from ATI, and integrated I/O cards may also have a bus mouse connector. Some bus mice are known as `InPort mouse'.

Note that some mouse manufactures have sold a package including a serial mouse and a serial interface card. Don't confuse this type of products with the genuine bus mouse.

PS/2 mouse

They are sometimes called `Mouse-port mouse'. The PS/2 mouse is becoming increasingly common and popular.

The PS/2 mouse is an intelligent device and may have more than three buttons and a wheel or a roller. The PS/2 mouse is usually compatible with the original PS/2 mouse from IBM immediately after power up. The PS/2 mouse with additional features requires a specialized initialization procedure to enable these features. Without proper initialization, it behaves as though it were an ordinary two or three button mouse.

Many mice nowadays can be used both as a serial mouse and as a PS/2 mouse. They has a logic to distinguish which interface it is connected to. However, the mouse which is not marketed as compatible with both serial and PS/2 mouse interface lacks this logic and cannot be used in such a way, even if you can find an appropriate adapter with which you can connect the PS/2 mouse to a serial port or visa versa.

XFree86 now supports the mouse with a wheel, a roller or a knob. Its action is detected as the Z (third) axis motion of the mouse. As the X server or clients normally do not use the Z axis movement of the pointing device, a new configuration option, `ZAxisMapping`, is provided to assign the Z axis movement to another axis or a pair of buttons (see below).

3. OS Support for Mice

3.1. Summary of Supported Mouse Protocol Types

	Protocol Types				
OS platforms	serial protocols	PnP serial "Auto"	BusMouse protocol "BusMouse"	PS/2 protocol "PS/2"	Extended PS/2 protocols "xxxPS/2"
BSD/OS	Ok	?	?	?	?
FreeBSD	Ok	Ok	Ok	Ok	SP*1
FreeBSD(98)	Ok	?	Ok	NA	NA
Interactive Unix	Ok	NA	?*1	?*1	NA
Linux	Ok	Ok	Ok	Ok	Ok
Linux/98	Ok	?	Ok	NA	NA
LynxOS	Ok	NA	Ok	Ok	NA
NetBSD	Ok	Ok	Ok	SP*1	SP*1
NetBSD/pc98	Ok	?	Ok	NA	NA
OpenBSD	Ok	Ok	Ok	Ok*1	OK*1
OS/2	SP*2	SP*2	SP*2	SP*2	SP*2
SCO	Ok	?	SP*1	SP*1	NA
Solaris 2.x	Ok	NA*1	?*1	Ok	NA
SVR4	Ok	NA*1	SP*1	SP*1	NA
PANIX	Ok	?	SP*1	SP*1	NA

Ok: support is available, NA: not available, ?: untested or unknown.
SP: support is available in a different form

*1 Refer to the following sections for details.
*2 XFree86/OS2 will support any type of mouse that the OS supports, whether it is serial, bus mouse, or PnP type.

3.2. BSD/OS

No testing has been done with BSD/OS.

3.3. FreeBSD

FreeBSD supports the `"SysMouse"` protocol which must be specified when the `moused` daemon is running in versions 2.2.1 or later.

FreeBSD versions 2.2.5 or earlier do not support extended PS/2 mouse protocols (`"xxxPS/2"`). Always specify the `"PS/2"` protocol for any PS/2 mouse in these versions regardless of the brand of the mouse.

FreeBSD versions 2.2.6 or later include the kernel-level support for these mice. Specify the `"PS/2"` or `"Auto"` protocol and the X server will automatically make use of the kernel-level support. In fact, you may always specify `"Auto"` to any mouse in these versions unless the mouse is an old serial model which doesn't support PnP.

3.4. FreeBSD(98)

The PS/2 mouse is not supported.

3.5. Interactive Unix

The PnP serial mouse support (the `"Auto"` protocol) is not supported for the moment.

The bus mouse and PS/2 mouse should be supported by using the appropriate device drivers. Use `/dev/mouse` for the `"BusMouse"` protocol and `/dev/kdmouse` for the `"PS/2"` protocol. These protocols are untested but may work. Please send success/failure reports to <*michael.rohleder@stadt-frankfurt.de*>.

3.6. Linux

All protocol types should work.

3.7. Linux/98

The PS/2 mouse is not supported.

3.8. LynxOS

The PnP serial mouse support (the `"Auto"` protocol) is disabled in LynxOS, because of limited TTY device driver functionality.

3.9. NetBSD

NetBSD 1.3.x and former does not support extended PS/2 mouse protocols (`"xxxPS/2"`). The PS/2 mouse device driver `/dev/pms` emulates the bus mouse. Therefore, you should always specify the `"BusMouse"` protocol for any PS/2 mouse regardless of the brand of the mouse.

XFree86 3.3.4 supports the `"wsmouse"` protocol introduced in NetBSD 1.4 along with the wscons console driver. You need to run binaries compiled on NetBSD 1.4 to have support for it though. Use

`"/dev/wsmouse0"` for the device. Refer to the *wsmouse(4)* manual page for kernel configuration informations.

3.10. NetBSD/pc98

The PS/2 mouse is not supported.

3.11. OpenBSD

The raw PS/2 mouse device driver `/dev/psm0` uses the raw PS/2 mouse protocol.

OpenBSD 2.2 and earlier does not support extended PS/2 mouse protocols (`"xxxPS/2"`) . Therefore, you should specify the `"PS/2"` protocol for any PS/2 mouse regardless of the brand of the mouse.

OpenBSD 2.3 and later support all extended PS/2 mouse protocols. You can select the `"Auto"` protocol for PnP PS/2 mice or any specific extended (`"xxxPS/2"`) protocol for non PnP mice.

There is also a cooked PS/2 mouse device driver `/dev/pms0` which emulates the bus mouse. Specify the `"BusMouse"` protocol for any PS/2 mouse regardless of the brand of the mouse when using this device.

3.12. OS/2

XFree86/OS2 always uses the native mouse driver of the operating system and will support any type of pointer that the OS supports, whether it is serial, bus mouse, or PnP type. If the mouse works under Presentation Manager, it will also work under XFree86/OS2.

Always specify `"OSMouse"` as the protocol type.

3.13. SCO

The bus and PS/2 mouse are supported with the `"OSMouse"` protocol type.

The `"OSMouse"` may also be used with the serial mouse.

3.14. Solaris

Testing has been done with Solaris 2.5.1 and 2.6. Logitech and Microsoft bus mice have not been tested, but might work with the `/dev/logi` and `/dev/msm` devices. Standard 2 and 3 button PS/2 mice work with the `"PS/2"` protocol type and the `/dev/kdmouse` device. The PnP serial mouse support (the `"Auto"` protocol) has been tested and does not work.

3.15. SVR4

The bus and PS/2 mouse may be supported with the `"Xqueue"` protocol type.

The `"Xqueue"` may also be used with the serial mouse.

The PnP serial mouse support (the `"Auto"` protocol) is not tested.

TOC

3.16. PANIX

The PC/AT version of PANIX supports the bus and PS/2 mouse with the `"Xqueue"` protocol type. The PC-98 version of PANIX supports the bus mouse with the `"Xqueue"` protocol type.

Mouse Support in XFree86 **:** *OS Support for Mice*
Previous: *Supported Hardware*
Next: *Configuring Your Mouse*

4. Configuring Your Mouse

Before using the `XF86Setup` or `xf86config` programs to set up mouse configuration, you must identify the interface type, the device name and the protocol type of your mouse. Blindly trying every possible combination of mouse settings will lead you nowhere.

The first thing you need to know is the interface type of the mouse you are going to use. It can be determined by looking at the connector of the mouse. The serial mouse has a D-Sub female 9- or 25-pin connector. The bus mice have either a D-Sub male 9-pin connector or a round DIN 9-pin connector. The PS/2 mouse is equipped with a small, round DIN 6-pin connector. Some mice come with adapters with which the connector can be converted to another. If you are to use such an adapter, remember that the connector at the very end of the mouse/adapter pair is what matters.

The next thing to decide is a device node to use for the given interface. For the bus and PS/2 mice, there is little choice; your OS most possibly offers just one device node each for the bus mouse and PS/2 mouse. There may be more than one serial port to which the serial mouse can be attached.

The next step is to guess the appropriate protocol type for the mouse. The X server may be able to select a protocol type for the given mouse automatically in some cases. Otherwise, the user has to choose one manually. Follow the guidelines below.

Bus mouse

> The bus and InPort mice always use `"BusMouse"` protocol regardless of the brand of the mouse.
>
> Some OSs may allow you to specify `"Auto"` as the protocol type for the bus mouse.

PS/2 mouse

> The `"PS/2"` protocol should always be tried first for the PS/2 mouse regardless of the brand of the mouse. Any PS/2 mouse should work with this protocol type, although wheels and other additional features are unavailable in the X server.
>
> After verifying the mouse works with this protocol, you may choose to specify one of `"xxxPS/2"` protocols so that extra features are made available in the X server. However, support for these PS/2 mice assumes certain behavior of the underlying OS and may not always work as expected. Support for some PS/2 mouse models may be disabled all together for some OS platforms for this reason.
>
> Some OSs may allow you to specify `"Auto"` as the protocol type for the PS/2 mouse and the X server will automatically adjust itself.

Serial mouse

> The XFree86 server supports a wide range of mice, both old and new. If your mouse is of a

relatively new model, it may conform to the PnP COM device specification and the X server may be able to detect an appropriate protocol type for the mouse automatically.

Specify `"Auto"` as the protocol type and start the X server. If the mouse is not a PnP mouse, or the X server cannot determine a suitable protocol type, the server will print the following error message and abort.

```
xf86SetupMouse: Cannot determine the mouse protocol
```

If the X server generates the above error message, you need to manually specify a protocol type for your mouse. Choose one from the following list:

- `GlidePoint`
- `IntelliMouse`
- `Logictech`
- `Microsoft`
- `MMHittab`
- `MMSeries`
- `MouseMan`
- `MouseSystems`
- `ThinkingMouse`

When you choose, keep in mind the following rule of thumb:

1. `"Logitech"` protocol is for old serial mouse models from Logitech. Modern Logitech mice use either `"MouseMan"` or `"Microsoft"` protocol.
2. Most 2-button serial mice support the `"Microsoft"` protocol.
3. 3-button serial mice may work with the `"Mousesystems"` protocol. If it doesn't, it may work instead with the `"Microsoft"` protocol although the third (middle) button won't function. 3-button serial mice may also work with the `"Mouseman"` protocol under which the third button may function as expected.
4. 3-button serial mice may have a small switch at the bottom of the mouse to choose between ``MS'' and ``PC'', or ``2'' and ``3''. ``MS'' or ``2'' usually mean the `"Microsoft"` protocol. ``PC'' or ``3'' will choose the `"MouseSystems"` protocol.
5. If the serial mouse has a roller or a wheel, it may be compatible with the `"IntelliMouse"` protocol.
6. If the serial mouse has a roller or a wheel and it doesn't work with the `"IntelliMouse"` protocol, you have to use it as a regular 2- or 3-button serial mouse.

If the `"Auto"` protocol is specified and the mouse seems working, but you find that not all features of the mouse is available, that is because the X server does not have native support for that model of mouse and is using a ``compatible'' protocol according to PnP information.

If you suspect this is the case with your mouse, please send report to *<XFree86@XFree86.Org>*.

Standardized protocols

Mouse device drivers in your OS may use the standardized protocol regardless of the model or class of the mouse. For example, SVR4 systems may support `"Xqueue"` protocol. In FreeBSD the system mouse device `/dev/sysmouse` uses the `"SysMouse"` protocol. Please refer to the OS support section of this file for more information.

5. XF86Config Options

The following new options are available for the `Pointer` section of the `XF86Config` file.

5.1. Buttons

This option tells the X server the number of buttons on the mouse. Currently there is no reliable way to automatically detect the correct number. This option is the only means for the X server to obtain it. The default value is three.

Note that if you intend to assign Z axis movement to button events using the `ZAxisMapping` option below, you need to take account of those buttons into `N` too.

```
Buttons N
```

5.2. ZAxisMappping

This option maps the Z axis (wheel) motion to a pair of buttons or to another axis.

```
ZAxisMapping X
ZAxisMapping Y
ZAxisMapping N M
```

The first example will map the Z axis motion to the X axis motion. Whenever the user moves the wheel/roller, its movement is reported as the X axis motion. When the wheel/roller stays still, the real X axis motion is reported as is. The last example will map negative Z axis motion to the button `N` and positive Z axis motion to the button `M`. If this option is used and the buttons `N` or `M` actually exists in the mouse, their actions won't be detected by the X server.

Currently this option can not be set in the `XF86Setup` program. You need to edit the `XF86Config` file by hand to add this option.

5.3. Resolution

The following option will set the mouse device resolution to `N` counts per inch, if possible:

```
Resolution N
```

Not all mice and OSs can support this option. This option can be set in the `XF86Setup` program.

Previous: *Configuring Your Mouse*
Next: *Mouse Gallery*

TOC

6. Mouse Gallery

6.1. MS IntelliMouse (serial, PS/2)

This mouse has been supported since XFree86 3.3. However, support in 3.3.2 is slightly different; the wheel movement is recognized as the Z axis motion. This behavior is not compatible with XFree86 3.3, but is more consistent with the support for other mice with wheels or rollers. If you want to make the wheel behave like before, you can use the new option `"ZAxisMapping"` as described above.

IntelliMouse supports the PnP COM device specification.

To use this mouse as a serial device:

```
Protocol "Auto" or "IntelliMouse"
Device   "/dev/xxxx" (where xxxx is a serial port)
```

To use this mouse as the PS/2 device and the OS supports PS/2 mouse initialization:

```
Protocol "IMPS/2"
Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device but the OS does not support PS/2 mouse initialization (the wheel won't work in this case):

```
Protocol "PS/2"
Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device and the OS supports automatic PS/2 mouse detection:

```
Protocol "Auto"
Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

6.2. Kensington Thinking Mouse (serial, PS/2)

This mouse has four buttons. Thinking Mouse supports the PnP COM device specification.

To use this mouse as a serial device:

```
Protocol "Auto" or "ThinkingMouse"
Device   "/dev/xxxx" (where xxxx is a serial port)
```

To use this mouse as the PS/2 device and the OS supports PS/2 mouse initialization:

```
Protocol "ThinkingMousePS/2"
Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device but the OS does not support PS/2 mouse initialization (the third and the fourth buttons act as though they were the first and the second buttons):


```
        Protocol "PS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device and the OS supports automatic PS/2 mouse detection:

```
        Protocol "Auto"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

6.3. Genius NetScroll (PS/2)

This mouse has four buttons and a roller. The roller movement is recognized as the Z axis motion.

To use this mouse as the PS/2 device and the OS supports PS/2 mouse initialization:

```
        Protocol "NetScrollPS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device but the OS does not support PS/2 mouse initialization (the roller and the fourth button won't work):

```
        Protocol "PS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device and the OS supports automatic PS/2 mouse detection:

```
        Protocol "Auto"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

6.4. Genius NetMouse and NetMouse Pro (serial, PS/2)

These mice have a "magic button" which is used like a wheel or a roller. The "magic button" action is recognized as the Z axis motion. NetMouse Pro is identical to NetMouse except that it has the third button on the left hand side.

NetMouse and NetMouse Pro support the PnP COM device specification. When used as a serial mouse, they are compatible with MS IntelliMouse.

To use these mice as a serial device:

```
        Protocol "Auto" or "IntelliMouse"
        Device   "/dev/xxxx" (where xxxx is a serial port)
```

To use this mouse as the PS/2 device and the OS supports PS/2 mouse initialization:

```
        Protocol "NetMousePS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device but the OS does not support PS/2 mouse initialization (the "magic button" and the third button won't work):

```
        Protocol "PS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device and the OS supports automatic PS/2 mouse detection:

```
        Protocol "Auto"
```

```
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

6.5. ALPS GlidePoint (serial, PS/2)

The serial version of this pad device has been supported since XFree86 3.2. `Tapping' action is interpreted as the fourth button press. (IMHO, the fourth button of GlidePoint should always be mapped to the first button in order to make this pad behave like the other pad products.)

To use this pad as a serial device:

```
        Protocol "GlidePoint"
        Device   "/dev/xxxx" (where xxxx is a serial port)
```

To use this mouse as the PS/2 device and the OS supports PS/2 mouse initialization:

```
        Protocol "GlidePointPS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device but the OS does not support PS/2 mouse initialization:

```
        Protocol "PS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device and the OS supports automatic PS/2 mouse detection:

```
        Protocol "Auto"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

6.6. ASCII MieMouse (serial, PS/2)

This mouse appears to be OEM from Genius. Although its shape is quite different, it works like Genius NetMouse Pro. This mouse has a "knob" which is used like a wheel or a roller. The "knob" action is recognized as the Z axis motion.

MieMouse supports the PnP COM device specification. When used as a serial mouse, it is compatible with MS IntelliMouse.

To use this mouse as a serial device:

```
        Protocol "Auto" or "IntelliMouse"
        Device   "/dev/xxxx" (where xxxx is a serial port)
```

To use this mouse as the PS/2 device and the OS supports PS/2 mouse initialization:

```
        Protocol "NetMousePS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device but the OS does not support PS/2 mouse initialization (the knob and the third button won't work):

```
        Protocol "PS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device and the OS supports automatic PS/2 mouse detection:

```
        Protocol "Auto"
```

```
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

6.7. Logitech MouseMan+ and FirstMouse+ (serial, PS/2)

MouseMan+ has two buttons on top, one side button and a roller. FirstMouse+ has two buttons and a roller. The roller movement is recognized as the Z axis motion. The roller also acts as the third button. The side button is recognized as the fourth button.

MouseMan+ and FirstMouse+ support the PnP COM device specification. They have MS IntelliMouse compatible mode when used as a serial mouse.

To use these mice as a serial device:

```
        Protocol "Auto" or "IntelliMouse"
        Device   "/dev/xxxx" (where xxxx is a serial port)
```

To use this mouse as the PS/2 device and the OS supports PS/2 mouse initialization:

```
        Protocol "MouseManPlusPS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device but the OS does not support PS/2 mouse initialization (the wheel and the fourth button won't work):

```
        Protocol "PS/2"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)
```

To use this mouse as the PS/2 device and the OS supports automatic PS/2 mouse detection:

```
        Protocol "Auto"
        Device   "/dev/xxxx"  (where xxxx is the PS/2 mouse device)

$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/mouse.sgml,v 1.1.2.12 1999/06/30
13:00:29 hohndel Exp $
```

Mouse Support in XFree86 **:** *Mouse Gallery*
Previous: *XF86Config Options*
Next: *Mouse Support in XFree86*

Information about the XInput extension in XFree86[tm]

TOC

Frédéric Lepied

2 November 1998

1. Introduction

2. Description

3. XFree86 implementation

3.1. Server side

3.2. Clients

1. Introduction

This document provides some information about the XInput extension implemented in XFree86[tm].

If you have any suggestions, comments, fixes or ideas regarding the XInput extension in XFree86[tm] or about this document, send e-mail to

lepied@XFree86.Org

Bug Reports should be sent to

XFree86@XFree86.Org

Questions or anything else should be posted to the NewsGroup

comp.windows.x.i386unix

2. Description

The XInput extension is a standard X Consortium extension. The goal of this extension is to allow additional input devices management to the X Window System. The documentation could be found in the X Consortium distribution under *xc/doc/hardcopy/Xi*.

3. XFree86 implementation

The XFree86 implementation contains 2 parts : the server part and two clients (*xsetpointer* and *xsetmode*).

3.1. Server side

The server supports the following extended devices :

- Joystick (only on supported systems ie. Linux, FreeBSD and NetBSD). Features :
  - Relative mode.
  - 2 valuators (x and y axis).
  - 2 buttons.
- Elographics touchscreen. Features :
  - Absolute mode.
  - 2 valuators (x and y axis).
  - 1 button.
- Stylus on MicroTouch touchscreen. Features :
  - Absolute mode.
  - 2 valuators (x and y axis).
  - 1 button.
- Finger on MicroTouch touchscreen. Features :
  - Absolute mode.
  - 2 valuators (x and y axis).
  - 1 button.
- Mouse. Features :
  - Relative mode.
  - 2 valuators (x and y axis).
  - up to 4 buttons.
- Wacom stylus. Features :
  - Absolute or relative modes.
  - 6 valuators :
    1. X axis.
    2. Y axis.
    3. pressure.
    4. X tilt.
    5. Y tilt.
    6. wheel.
  - 3 buttons.
  - Proximity report.
  - Motion history capability.


TOC

    - Macro/function buttons are reported as keys.
- Wacom eraser. Features :
    - Absolute or relative modes.
    - 6 valuators :
        1. X axis.
        2. Y axis.
        3. pressure.
        4. X tilt.
        5. Y tilt.
        6. wheel.
    - 1 button.
    - Proximity report.
    - Motion history capability.
    - Macro/function buttons are reported as keys.
- Wacom cursor. Features :
    - Absolute or relative modes.
    - 6 valuators :
        1. X axis.
        2. Y axis.
        3. pressure.
        4. X tilt.
        5. Y tilt.
        6. wheel.
    - 16 buttons.
    - Proximity report.
    - Motion history capability.
    - Macro/function buttons are reported as keys.
- SummaSketch tablet. Features :
    - Absolute or relative modes.
    - 2 valuators (x and y axis).
    - 2 buttons stylus or 4 buttons puck.
    - Proximity report.
    - Motion history capability.
- AceCad tablet. Features :
    - Absolute or relative modes.
    - 2 valuators (x and y axis).
    - 2 buttons stylus or 4 buttons puck.
    - Proximity report.
    - Motion history capability.
- Calcomp DrawingBoard tablet. Features :
    - Absolute or relative modes.
    - 2 valuators (x and y axis).
    - 4 buttons stylus or 16 buttons puck.

- Proximity report.
- Motion history capability.

- SWITCH virtual device. Features :
  - Absolute mode.
  - 1 valuator (device id) which reports the id of the device controlling the core pointer (works with the AlwaysCore feature see bellow).
- SGI button box. Features :
  - Absolute or relative modes.
  - 8 valuators.
  - 32 buttons.
  - Motion history capability.

To enable an extended device, you must add en entry in the *XF86Config* file. Consult to the *XF86Config* man pages to see the details.

The XFree86 implementation supports a non standard feature called *AlwaysCore* which enables an XInput device to send both core and extended events at the same time. To enable it you have to add the *AlwaysCore* keyword to the subsection describing your device in the *XF86Config* file. The *SWITCH* virtual device reports a Motion event when another device takes over the control of the core pointer. The id of the new device is reported in the first valuator of the event.

3.2. Clients

xsetpointer is used to change the device controlling the core pointer and to list available extended devices.

xsetmode is used to change the mode (absolute or relative) of an extended device. The device has to support relative and absolute modes and the device must not control the core pointer.

Consult the man pages for details.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/xinput.sgml,v 3.6.2.3 1999/06/30
13:00:30 hohndel Exp $
```

```
$XConsortium: xinput.sgml /main/3 1996/10/27 11:06:13 kaleb $
```

Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04

Takis Psarogiannakopoulos

July 27, 1999

1. Whats new
2. GENERAL:
3. Configuration for the build:
4. DISCUSSION ABOUT GCC
5. BUILD
6. INSTALLATION OF THE BINARY:
7. What is about:

1. Whats new

July 27, 1999

DG has fix the streams bug in /usr/lib/tcpip.so . (Read below) The workaround in the July 25 source code has been removed. I've been told from DG that BSD sockets perform better in DGUX than SVR4 native STREAMS. From R4.20MU06 DG/ux will have the correct tcpip.so lib (no bug in STREAMS). If you have MU05,MU04 and you want for some reason STREAMS definitely conntact DG for an upated /usr/lib/tcpip.so in /usr/lib (patch for your MU04,5).

DG/ux at the moment lacks the sysi86 syscall and the definition of SYSI86IOPL (that is in <sys/sysi86.h> but guarded by a UNIXWARE defintion that of course cannot be applicable to DG/ux). Until this header is accessible by a simple -DDGUX , and _sysi86, sysi86 subroutines added to libc (or some other extra library) we need to define the DG_NO_SYSI86 to be 1. If DG makes the above modifications , you will need to manually edit the files (before building!)
xc/programs/Xserver/hw/xfree86/SuperProbe/OS_DGUX.c
xc/programs/Xserver/hw/xfree86/etc/scanpci.c
xc/programs/Xserver/hw/xfree86/os-support/dgux/dgux_video.c and eliminate DG_NO_SYSI86 flag by changing DG_NO_SYSI86 1-->0.

July 25, 1999

A major bug has now been corrected in this release. According to this since the STREAMS interface of DG/ux were broken the server was listening not to port 6000 (= 0x1770) but to 0x7017. All binaries that you have from 3.3.3.1, 3.3.3 they will work locally (if you run them in the same machine) but NOT remotely because they use an Xlib that tries to connect to port 28365. If you want to run them remotely YOU MUST recompile them! Steve thank you for bringing this to my attension initially but was too bussy then to look at it in detail... Perhaps I should have...

We now use sockets intead of ioctls. But I 've fix and tested the STREAMS also.

David thanks for making me realize that this was indeed a problem in DG/ux.

I 've also take the trouble to port gdb-4.17/8 and ddd (X inter) in pub/XFree86/3.3.5/binaries/DGUX-ix86/GDB_BETA (both binaries and source code) in order to be able to send me traces of core files produced by Xservers. Try

```
gdb /usr/X11R6/bin/X location of core/core
gdb: where
```

and send me what you see. (I suppose that the Xserver executable is in /usr/X11R6.3/bin)

From 3.3.3.1: Several bugs are now fixed. DG/ux support is on this official patch. All configuration xc/config/cf/DGUX.cf and in xf86site.def. Also Imake autodetects (thanks to David Dawes) the DGUX OS (including Release version). So only a simple "make World" is needed anymore to build in ix86 DG/ux. Usually the process to build is (after unpacking and patching the source "xc" tree) to go to xc/config/cf copy the file xf86site.def to site.def and edit the files DGUX.cf ,site.def for whatever changes you need to do.

Default ProjectRoot in 3.3.5 is /usr/X11R6 (except if you set this specifically in DGUX.cf either as /usr/X11R6.3-- --my choice, or in whatever you like).

From: December 1, 1998 IMPORTANT: PLEASE READ THE FILE README-GCC-2.8.1 GCC VERSION 2.8.1 is recommended for the DGUX build of X11R6.3 You can ignore below the _old_ converstation about gcc compiler if you already run a gcc-2.8.1 in your machine.

Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04 **:** *Whats new*
Previous: *Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04*
Next: *GENERAL:*

2. GENERAL:

Get from ftp.xfree86.org the XFree 3.3.x source code:

```
ftp ftp.xfree86.org
login: ftp
passwd: your e-mail address
cd pub/XFree86/3.3.5/source
You need the files:
X335src-1.tgz
X335src-2.tgz
X335src-3.tgz
```

Get also the contributed X software

```
cd /pub/XFree86/3.3.5/source
X335contrib.tgz
```

If you have the source tarballs of 3.3.3 go to /pub/XfFree86/3.3.5/binaries/DGUX-ix86/SOURCE get the DGUX source patch

```
3.3.3-3.3.5-DGUX.diff.gz
X335contrib-DGUX.diff.gz (patch for the contrib software).
```

to avoid downloading all the source code again.

Sorry some DGUX changes they didnt make in time for the official release date of 3.3.4... I do this for free, so some times I may be excused for something like this...

To build X11R6.3 you need also the tools for DG/ux (see discussion below). They will be in ftp dpmms.cam.ac.uk (University of Cambridge,Department of Pure Mathematics) in /pub/takis/DGUX-Tools/BuildXtools.tar.gz (anonymous ftp) or in the ftp.xfree86.org ... (pub/XFree86/3.3.5/binaries/DGUX-ix86/SOURCE/BUILD-TOOLS/ BuildXtools.tar.gz).

Using a big filesytem (I use a virtual disk "xf86work" mounted on /xf86work of size 1400000 blocks) copy the source as:

```
cp X335src-1.tgz  /xf86work/X335src-1.tar.gz
cp X335src-2.tgz  /xf86work/X335src-2.tar.gz
cp X335src-3.tgz  /xf86work/X335src-3.tar.gz
```

(or X333src-1,2,3.tar.gz as above plus the DG/ux patch to 3.3.5 i.e. the file 3.3.3-3.3.5-DGUX.diff.gz)

And maybe the contributed software:


```
cp  X335contrib.tgz           /xf86work

(cp  X335contrib-DGUX.diff.gz  /xf86work).
```

If you get the patches from DGUX-ix86, to build from source 3.3.3 your first problem is that in order to apply the source patch you need "patch" a very common GNU program that DG/ux doesnt have!? This program is in BuildXtools.tar.gz (or ftp prep.ai.mit.edu cd/pub/gnu get patch-2.5.tar.gz) So lets speak about these tools before anything else: Using "sysadm" mount a filesystem usr_local under /usr/local with size 200000 blocks. Then copy the BuildXtools file in /usr and give:

```
gzip -d < BuildXtools.tar.gz | tar xvf -
```

It will unfold in the new filesytem /usr/local. Then go to your ".profile" (in your Home dir , mine eg is /admin) and add the /usr/local/bin in your path and the /usr/local/lib in your lib-path.

What you need is like that: (vi .profile)

```
PATH=/usr/local/bin:/sbin:/usr/sbin:/usr/bin
if [ -d /usr/opt/X11/bin ]
then
  PATH=$PATH:/usr/opt/X11/bin
fi
export PATH
(/usr/local/bin is before any other path!)

LD_LIBRARY_PATH=/usr/local/lib:/usr/lib:/usr/ccs/lib
export LD_LIBRARY_PATH
```

then exit and re-login so that your new modified .profile will take effect.

Now do the following:

```
cd /usr/sbin
cp install install_dg
rm install
cp /usr/local/bin
cp install /usr/bin  *(make the GNU install the default install)*
(DG/ux install is useless)
(Look also the file xc/config/cf/DGUX.cf , below).

(Or get GNU make-3.77 and copy install.sh (or -sh) to a /usr/bin/install)

cd /usr/bin
cp true    /usr/local/bin
cd /usr/local/bin
ln -s true ranlib (Install true as ranlib in the DG/ux system)
```

Usually giving -v,-V or --version will give you build info on all the tools in /usr/local/bin (try it).

Look the discussion for gcc before you build (below)


TOC

Now untarr the source tree: In /xf86work (or whatever name you gave to the big filesystem for the build)

```
gzip -d < X335src-1.tar.gz | tar xvf -
gzip -d < X335src-2.tar.gz | tar xvf -
gzip -d < X335src-3.tar.gz | tar xvf -
```

(If you have X-3.3.3 do the same for the 3.3.3 sources and then apply the 3.3.5-DGUX source patch

```
gzip -d < 3.3.3-3.3.5-DGUX.diff.gz | patch -p0 -E).
```

The directory xc in now present in your build filesystem.

Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04 : *GENERAL:*
Previous: *Whats new*
Next: *Configuration for the build:*

3. Configuration for the build:

Almost all you need is in "DGUX.cf" located in xc/config/cf. Edit the file DGUX.cf and site.def and change what ever you need. Remember DGUX.cf overwrites site.def. The default ProjectRoot for XFree86-3.3.5 is now /usr/X11R6 (located in site.def). If you want to change this to whatever you like (I prefer /usr/X11R6.3 and a link in /usr X11R6->X11R6.3) edit DGUX.cf and locate the entry:

```
#if 0
#define ProjectRoot /usr/X11R6.3
#endif
```

Eliminate the #if 0 , #endif. Then change this to whatever you prefer. (I prefer the above for the precompiled binaries)

The DG/ux malloc is crap and keeps bringing problems with some X software so I dont use it. Instead there is a port of GNU malloc in /usr/local (it came with the BuildXtools file). Dont try to build with the /lib/libmalloc.a of DG/ux and then send me e-mails that some programs they dont work properly. In my build I use tcl8.0 and tk8.0 since the xconfig of R4.20MU03 is reporting incorrect values for the monitors and XF86Setup need to be build in order to manage to adjust the display. If you dont have this (or dont want this) comment out the lines about tcl,tk, (in DGUX.cf)

```
/*******TCL TK DEFINITIONS ***********/
 #define HasTk  YES-->NO
 ...
 #define HasTcl YES-->NO
```

Also 'GNU make' is required for the correct X11R6.3 build. (it is in Buildxtools file). If you decide yes to tcl,tk obtain the files

```
tcl8.0.3.tar.gz
tk8.0.3.tar.gz    (from some ftp)
```

(or newer versions) and compile them before the building of X11R6.3 (Build first tcl8.0.3 then tk8.0.3).

4. DISCUSSION ABOUT GCC

There are so much things that I can say for the system gcc of DG/ux. If I was keeping track for the programs that fail using this compiler I will certainly have fill a book (conveniently for the DG of course). But my work is not to correct bugs for the DG/ux compiler or anything else) , and in particular to collect reports for the genius of DG. (DG:Sorry guys nothing personal. I am a pure Mathematician , I am doing all this work for pleasure, I dont want any money from DG or anybody else , I am not looking to become a employer of DG,and I am NOT a trouble shooter of the DG/ux in general. But maybe some times if you help I may be able to help you also).

What I wanted to do is to build X11. Thats why you will find in BuildXtools a new gcc. This gcc is build for DG/ux R4.20MU02. so you have to upgrade your DG/ux OS version to the above. But it is solid to build not only X11 but whatever else you want. DO NOT use gcc of DG/ux. If you do I cannot tell you anything about any problems that you have. To complete the installation of this new gcc do the following:

```
cp -r /usr/local/gcc-dgux  /usr/opt/sdk/sde/ix86dgux/usr/lib
cd  /usr/opt/sdk/sde/ix86dgux/usr/lib
rm gcc
ln -s gcc-dgux gcc   (set link gcc--->gcc-dgux)

cd /usr/local
cp -r /usr/local/gcc-dgux  /usr/sde/ix86dgux/usr/lib
cd  /usr/sde/ix86dgux/usr/lib
rm gcc
ln -s gcc-dgux gcc  (set link gcc-->gcc-dgux)
```

To come back to your old DG/ux gcc just change the above two links gcc-->gcc-dgux to gcc-->gcc-2 with the command: (in both the above two dirs)

```
rm gcc
ln -s gcc-2 gcc
```

/usr/bin/gcc -v should report the version that you have. To build succesfully this version of X11 gcc is a *MUST*.

Dynamic loading Servers: Edit xc/config/cf/DGUX.cf and change the entry

#ifndef BuildDynamicLoading #define BuildDynamicLoading NO --->YES. #endif

Remeber when you build you will see lots of errors and the servers will NOT build! This is because the

dynamic linker doesnt know the locatios of the newly created R6 libX's. So after the (seem faulty) building do a

make DESTDIR=ProjectRoot/lib install

(look below for install, ProjectRoot the location that you choose in the file xc/config/cf/DGUX.cf above)

So that all your new libXR6 libraries will go there. (do a cd ProjectRoot/lib to make sure).

Then go to your home dir and declare the path ProjectRoot/lib dir in you LD_LIBRARY_PATH (your profile) as:

LD_LIBRARY_PATH=ProjectRoot/lib:$LD_LIBRARY_PATH export LD_LIBRARY_PATH

Then relogin!

Now just _rebuild_ A FULL XFree86-3.3.5 with the entry

#define BuildDynamicLoading YES in your DGUX.cf.

This time you will build _all_ XFree86-3.3.5 correctly.

Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04 **:**
DISCUSSION ABOUT GCC
Previous: *Configuration for the build:*
Next: *BUILD*

5. BUILD

In the usual X11R5 of DG (mwm) open an xterm and give: (/bin/sh = Bourne shell)

```
cd xc
make World > Build-dg.log
```

In that way you will get all the error meggages in your screen. Dont worry with messages about -znodefs.

Note: the old command

```
make World BOOTSTRAPCFLAGS="-DDGUX" > Build-dg.log
```

is no longer needed since imake will detect the DGUX OS (in Version => 3.3.3.1) and set up things. However this command will also work.

And in another xterm give cd xc tail -f Build-dg.log to watch the building progress.

To install X11R6.3 XFree version 3.3.5 after the build you must have a filesystem say usr_X11R6.3 mounted to the directory /usr/X11R6 size 350000 blocks. (or whatever you choose to be your ProjectRoot, if you modify DGUX.cf for another ProjectRoot than the default /usr/X11R6 in site.def). Then give

```
make install
make install.man (install the man pages)
```

The installation will not put an XF86Config in your /usr/X11R6.3/lib/X11/ and so if you give startx the new X11 will not start. Read the file README-X3331.DGUX in this ftp site (located in the binaries) about the whole installation procedure of X11R6.3. Or quickly you can do: (I remind: DG/ux mouse device "/dev/mouse") cd /usr/X11R6.3/bin ln -s XF86_VGA16 X Then put in your .profile the path /usr/X11R6.3/bin and run the XF86Setup program. Ajust first the mouse device then everything else. (You need to read really the file README-DGUX.INSTALL-> look in the end of this file).

To build the contributed software with XFree86-3.3.5 get the X335contrib.tgz and do

```
gzip -d < X335contrib.tgz | tar xvf -
```

(Or for 3.3.3 sources unpack X333contrib.tgz and apply the DGUX patch as

```
gzip -d < X335contrib-DGUX.diff.gz  | patch -p0 -E ).
```

Please note: You must have already install and active the X11R6.3 that you build so that the imake is working properly for your system. Read below for how to install this Xwindow system. After that you

could do:

```
cd contrib
xmkmf -a
make
make install
make install.man (for installing the man pages)
```

Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04 : *BUILD*
Previous: *DISCUSSION ABOUT GCC*
Next: *INSTALLATION OF THE BINARY:*

6. INSTALLATION OF THE BINARY:

NOTE:This executable has been compiled with the macro -DPENTIUM_CHANGE (that all the new Aviion machines support). If you have an old i486 (rather unlikely) the executable will NOT RUN correctly. But we havent use -mcpu=pentiumpro, so the executable will work on ALL PENTIUM machines.

- About Project Root: I choose as ProjectRoot for ix86 DG/ux the location /usr/X11R6.3. The default (in 3.3.5 sources) is the /usr/X11R6. To cover this we make a link in /usr as X11R6->X11R6.3 (read below) ;so dont forget to do this link. I dont like the location /usr/opt/X11 (default location of DG X11) that some of you have suggested to me, I believe it is a good idea to keep the original X11 as is for several reasons.
- Make a filesystem,using sysadm, mounted under "/usr/X11R6.3". This is the default location of X11R6.3 , you cannot change this except if you recompile the whole source of X. (Please dont send e-mails about this). The size of this filesystem should be around 175 MB(350000 blocks). The list of files is:

```
X3353DL.tgz    3D_Labs XServer ... etc
X3358514.tgz
X335AGX.tgz
X335I128.tgz
X335Ma32.tgz   Mach32 Xserver
X335Ma64.tgz   Mach64 Xserver
X335Ma8.tgz
X335Mono.tgz
X335P9K.tgz
X335S3.tgz
X335S3V.tgz
X335SVGA.tgz   SuperVGA Xserver (Supports AV3700 Cirrus)
X335VG16.tgz   VGA16 Xserver (needed by XF86Setup)
X335W32.tgz
X335bin.tgz    BIN (you must have this)
X335cfg.tgz
X335doc.tgz
X335f100.tgz
X335fcyr.tgz
X335fnon.tgz
X335fnts.tgz
X335fscl.tgz
```


```
X335fsrv.tgz
X335lib.tgz    LIB (you must have this)
X335lkit.tgz   Linkkit (X development)
X335man.tgz    Man pages
X335nest.tgz
X335prog.tgz
X335prt.tgz
X335set.tgz
X335vfb.tgz
preinst.sh    Install script
extract        The XFree86 extract program (for ix86 DG/ux)

SUMS.md5       CheckSums for the integrity of the files
```

(Try compile the GNU textutils-1.22.tar.gz from prep.ai.mit.edu /pub/gnu. md5sum is there).

You need at least:

```
X335bin.tgz
X335lib.tgz
```

And the correct Xserver for your machine/Graphics card. In my opinion take all files , in the future you may need to switch to another graphics device etc ... (mget *). Generally it is good to have the full distribution of the X11R6.3 window system ,it should make life easier in DG/ux.

(Trivial:you must have root privilege).

- Unpack the *.tgz files in your / so that it will go directly inside to the new filesystem /usr/X11R6.3. After you do that cd /usr and do a link : ln -s X11R6.3 X11R6. (Use the install script). This link will indicate in XF86 programs like XF86Setup where the new X11 window system is.
- cd your home dir and backup your .profile as "cp .profile myprofile". Then cd /usr/X11R6.3. Copy the file HOME.profile-X11R6.3 to your home dir as "cp HOME.profile-X11R6.3 your home dir/.profile" ,then cd your home dir and "chmod 644 .profile" to make sure that the new profile is active. This is because you need to tell to the system to look for the X software in a different location than the usual /usr/bin/X11 of DG/ux. Also you need to tell to the system that the new X libraries are in /usr/X11R6.3/lib.

  You NEED to re-login in order to make the new .profile active ! ** DO NOT GIVE "startx" AFTER THAT, READ Configuration below !**
- About Configuration: DG/ux has a program (actually a script) called xconfig that makes the configuration for you. Usually when you run xconfig in the DG/ux-X11R5 it creates a file XdgConfig in /var/X11/Xserver which is the corresponding of the XF86-configuration file located in /usr/X11R6.3/lib/X11/XF86Config. This file ,in the section monitor, has all values for your monitor. Please Note: Unfortunately in DG/ux R4.20MU02 things change. Instead of going forwards we going backwards... xconfig reports crazy values for my DG-DA1765VA monitor. So if you have a CDROM of DG/ux R4.11MU02,or MU03 use it to find an xconfig that will give you reliable values for your monitor.

TOC

Your best bet is to use XF86Setup for correct adjustments.

Thats the reason that in this binary there is a mininum tcl,tk(version 8.0). Before you run XF86Setup read the relevant documents found in www.xfree86.org. (Or read below for a hand-made configuration).

Notice about XF86Setup: You will will see the message "The program is running on a different virtual" "Please switch to the correct virtual terminal"

DG/ux does NOT have any virtual terminals. But XF86Setup uses a script that doesnt checks for this. So it is printing this message anyway. Ignore it and dont send e-mails asking how to set the virtual terminal! XF86Setup WORKS for SURE (if you use it correctly) to set your configuration. Just remember:

1. make a link in /usr/X11R6.3/bin: ln -s XF86_VGA16 X
2. set mouse device in your XF86Config to /dev/mouse (this the mouse in DGUX)
3. set the correct mouse protocol. (usually for a typical AViiON PS/2).

Or just cd /usr/X11R6.3/lib/X11 and copy XF86Config.eg.dgux to XF86Config (it is for a PS/2 protocol mouse that almost all AViiON's have) , then run XF86Setup and choose to use this XF86Config file as default (so mouse works).

Alternatively, you can run xf86config, a non-graphical configuration utility but you will need to enter manually the values for your monitor. If you have the small booklet that came with the monitor they are inside.

Hand made configuration: I have an DG/ux Medium resolution (1280x1024) 17 inch DG-26059,DA1765VA. **ONLY IF YOU HAVE THE ***EXACT SAME*** MONITOR USE THE FILE XF86Config_SVGA_DGUX that you will find in /usr/X11R6.3/. IT IS IN YOUR OWN RISK IF YOU DECIDE TO USE THIS FILE WHEN YOU DONT HAVE THE SAME MOTINOR AS MINE. YOU CAN DAMAGE YOUR VIDEO MONITOR OR YOUR GRAPHICS CARD.**

An examble of how to use the Accel Servers (eg ATI=XF86_Mach64) is in the file XF86Config_ATI_DGUX. Again remember: I have an DG/ux Medium resolution (1280x1024) 17 inch DG-26059,DA1765VA. **ONLY IF YOU HAVE THE ***EXACT SAME*** MONITOR USE THE FILE XF86Config_ATI_DGUX that you will find in /usr/X11R6.3/. IT IS IN YOUR OWN RISK IF YOU DECIDE TO USE THIS FILE WHEN YOU DONT HAVE THE SAME MOTINOR AS MINE. YOU CAN DAMAGE YOUR VIDEO MONITOR.**

Start with the file XF86Config.eg as a prototype. READ the REAME.Config . In Cirrus chips you need to read the file README.cirrus located in /usr/X11R6.3 There is a problem with the accelarated XAA code ,so you need to try to put the following option in your XF86Config:

```
Option "no_mmio" (in Section Screen ,subsection display).
```

Look in the XF86Config_SVGA_DGUX to see how this can be done . If this doesnt work (it will probably) try Option "noaccel" or "no_bitblt".Again READ the file README.cirrus (and README.Config). I suggest to print (in paper) the file XdgConfig and have a look in it. Then it should be quite trivial to figure out what you have to do with the XF86 file ie XF86Config in the

sections mouse, keyboard, screen ... After you have a correct XF86Config in /usr/X11R6.3/lib give

```
chmod 444 XF86Config.
```

Supposing that you have already re-login so that the new .profile is active and you have the correct XF86Config file (as your XdgConfig suggest) (DO not forget for a cirrus to put the Option "no_mmio" in section screen !), give startx and the new X11 will start . Remember: You can shut down at any point the Xserver by pressing CONTROL+ALT+BACKSPACE (if something goes wrong). Also Xservers dont produce messages unless to want them to do so. This is because the DG/ux console driver some times causes corruption of the screen if you print text during the startup of the Xserver. If you require messages try in bash shell to give: (bash#)

```
X -verbose >& info1  or even
X -verbose -verbose >& info2 for more messages.
```

Then when the server is up press CONTROL+ALT+BACKSPACE to shutdown the Xserver. File info1 (or info2) have all relevant info about your graphics card , display memory etc ... I suggest you do that at least one time before start using the new X11R6.3. Read this info file to see if all ok. If not try change settings in your XF86Config to make thinks correct.

If you have an ATI Rage II (or RageII+) use the server XF86_Mach64 (make a link link X--->XF86_MACH64, or run xf86config, or use XF86Setup above).

- If you want to compile programs with the X11R6.3 the headers in /usr/include /X11 pointing to /usr/opt/X11 of DG/ux is a problem . Do:

a): unmounting the /usr/opt/X11 will prevent the sysadm to use the X graphical interface. But this will be the only thing that you loose. The correct thing to do for X11R6.3 is to delete the filesystem /usr/opt/X11 and make a link /usr/opt/X11--->/usr/X11R6.3 , so that the libraries from dglib and /usr/lib point correctly to the new ones in /usr/X11r6.3/lib. Before you unmount this filesystem you need to do this:

```
cd /usr/opt/X11/include
cp -r Mrm    /usr/X11R6.3/include
cp -r uil    /usr/X11R6.3/include
cp -r Xm     /usr/X11R6.3/include
cd /usr/X11R6.3/include
ln -s uil Uil
```

LIBRARIES:

```
and cd /usr/opt/X11/lib
cp libXm.a       /usr/X11R6.3/lib
```

and similarly copy the following libraries:

```
libX11.so.2, libX11.so.5, libXIM.so.1, libXaw.so.1, libXaw.so.2,
```


```
libXext.so.2, libXi.so.2, libXimp.so.1, libXm.so.2, libXmu.so.
libXsess.so.1, libXsi.so.1, libXt.so.2, libXt.so.5.0, libMrm.a,
libUil.a, libX11_s, libXR4sco_s
```

into /usr/X11R6.3/lib.

Then cd /usr/X11R6.3/lib and make links:

```
ln -s libXm.so.2 libXm.so
ln -s libXm.so.2 libXm.so.1
ln -s libXm.so.2 libXm.so.5.0

ln -s libX11.so.5.0 libX11.so.1

ln -s libXIM.so.1 libXIM.so.5.0

ln -s libXaw.so.2 libXaw.so.5.0

ln -s libXext.so.2 libXext.so.5.0
ln -s libXext.so.2 libXext.so.1

ln -s libXi.so.2 libXi.so.1
ln -s libXi.so.2 libXi.so.5.0

ln -s libXimp.so.1 libXimp.so.5.0

ln -s libXmu.so.2 libXmu.so.5.0

ln -s libXt.so.2 libXt.so.5.0

ln -s libXsi.so.1 libXi.so.5.0

cd /usr/X11R6.3/lib
rm libXmu.so (to avoid undefs when building X software)
```

Also you need to correct the links in /usr/dglib at least! (the correct thing to do is modify also /usr/lib links to /usr/opt/X11 libs). Try

```
cd /usr/
tar -cvf dglib-orig.tar dglib
gzip dglib-orig.tar
```

(to minimize the space dglib-backup takes) then

```
cd /usr/dglib
```

and delete ALL links to libraries in /usr/opt/X11. Then copy the script create_new_links_in_dglib

(found in /usr/X11R6.3 to /usr/dglib and cd /usr/dglib execute script. This will create all new l with the X11R6.3 X window system.

But remember to do in the end :

```
cd /usr/dglib
rm *.a (no static libs links in dglib)
```

Then unmount (delete) the old X11 by giving "umount /usr/opt/X11".

NOTE: If you compile programs in the X11R6 make sure that you unmount /usr/opt/X11 or you eliminate the links in /usr/lib to the OLD libX's in /usr/opt/X11/lib. Otherwise gcc will link these old libraries! and the binary will not run correctly. Always after an R6 compilation do "ldd prog" to make sure that the binary loads only R6 version libraries (except maybe the motif library libXm.so.2), --if you dont use the static libXm.a

b:)

```
cd /usr/include
tar -cvf old-X11headers.tar X11
gzip old-X11headers.tar
```

so that you store your old headers in /usr/include.

Then cd /usr/include/ and delete

```
      rm -r X11
      rm Xm
      rm Mrm
      rm Uil
      rm uil
Make new links as:
cd /usr/include
ln -s ../X11R6.3/include/X11 X11
ln -s ../X11R6.3/include/uil Uil
ln -s ../X11R6.3/include/uil uil
ln -s ../X11R6.3/include/Xm  Xm
ln -s ../X11R6.3/include/Mrm Mrm
```

Instructions for Building XFree86 on an Intel Pentium Aviion machine with DG/ux R4.20MU04 :
INSTALLATION OF THE BINARY:
Previous: *BUILD*
Next: *What is about:*

TOC

7. What is about:

This new X11R6 are not simply an upgrade of the servers to the latest ones. It is a new programming platform in your DG/ux system to allow you to import all this *FREE* or not software for the X window system. This software will not compile in the old (and ungly) X11 of Data General. The imake command that is implemented in almost all the (source) software for X11 (free or not) will not work with the totally broken "imake" command of /usr/opt/X11 of DG/ux.

The imake of DG/ux X11R5 is badly broken: I have seen DG/ux releases R4.11,MU01, ...MU04, R420, R4.20MU02 ,R4.20MU03 and nobody bother to look in all these releases the imake command...

While until now DG was rather hostile to the prospect of a new X11 in DG/ux some new folks there they have turn their interest in X11R6 (XFree86) these days. That is good of cource because the ultimate target is to make XFree86 (3.3.5 or whatever version) to run in _their_ DG/ux Unix! I will be able to make a much better X11 in DG/ux if I could had some access to DG/ux sources (i.e. the original R5 sources , but not only --eg kernel driver sources as for examble the DG/ux kernel console driver sources).

I have compile almost anything that runs for Linux in DG/ux using this X11. In doing this work in XFree86 I would like to express my thanks to David Dawes that he help me all the time with several techical questions. Also D.T. is one of the people that offer valuable help. Finally I want to express my thanks to John H. for enlight me in some syscall issues.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/DGux.sgml,v 1.1.2.3 1999/08/03
09:41:42 hohndel Exp $
```

README for XFree86 on FreeBSD

TOC

Rich Murphey, David Dawes

8 November 1998

1. What and Where is XFree86?
2. FreeBSD 3.0 and ELF
3. Installing The Display Manager (xdm)
4. Configuring X for Your Hardware
5. Running X
6. Rebuilding Kernels for X
7. Building X Clients
8. Thanks

1. What and Where is XFree86?

XFree86 is a port of X11R6.3 that supports several versions of Intel-based Unix. It is derived from X386 1.2, which was the X server distributed with X11R5. This release consists of many new features and performance improvements as well as many bug fixes.

For further details about this release, including installation instructions, please refer to the Release Notes.

See the Copyright Notice.

Binaries for XFree86 on FreeBSD 2.2.x and 3.0 are available from:

ftp://ftp.XFree86.org/pub/XFree86/current/binaries/

Send email to *Rich-Murphey@Rice.edu* or *XFree86@XFree86.org* if you have comments or suggestions about this file and we'll revise it.

2. FreeBSD 3.0 and ELF

The FreeBSD-3.0 binary distribution is ELF only. The Xbin.tgz tarball contains a.out libraries for compatibility purposes.

3. Installing The Display Manager (xdm)

The display manager makes your PC look like an X terminal. That is, it presents you with a login screen that runs under X.

The easiest way to automatically start the display manager on boot is to add a line in `/etc/ttys` to start it on one of the unoccupied virtual terminals:

```
ttyv4  "/usr/X11R6/bin/xdm -nodaemon" xterm    on secure
```

You should also make sure that `/usr/X11R6/bin/X` is a symbolic link to the Xserver that matches your video card or edit the file Xservers in `/usr/X11R6/lib/X11/xdm` to specify the pathname of the X server.

The change to `/etc/ttys` won't take effect until you either reboot or "`kill -HUP 1`" to force initd to reread `/etc/ttys`. You can also test the display manager manually by loging in as root on the console and typing "`xdm -nodaemon`".

4. Configuring X for Your Hardware

The `XF86Config` file tells the X server what kind of monitor, video card and mouse you have. You *must* create it to tell the server what specific hardware you have.

It is strongly recommended that you read through the QuickStart guide, and use either the `XF86Setup' utility (which requires the VGA16 server to be installed), or the `xf86config' utility to generate an XF86Config file.

When you run the `XF86Setup' utility, do NOT touch the mouse until you are finished with mouse set up. Otherwise, the VGA16 server and the mouse device driver may get confused and you may experience mouse and/or keyboard input problems.

If you are running ``mused'' (see the man page for `mused(8)`) in FreeBSD versions 2.2.1 or later, you MUST specify `SysMouse` as the mouse protocol type and `/dev/sysmouse` as the mouse device name, regardless of the brand and model of your mouse.

If you are NOT running ``mused'', you need to know the interface type of your mouse, `/dev` entry and the protocol type to use.

The interface type can be determined by looking at the connector of the mouse. The serial mouse has a D-Sub female 9- or 25-pin connector. The bus mouse has either a D-Sub male 9-pin connector or a round DIN 9-pin connector. The PS/2 mouse is equipped with a small, round DIN 6-pin connector. Some mice come with adapters with which the connector can be converted to another. If you are to use such an adapter, remember the connector at the very end of the mouse/adapter pair is what matters.

The next thing to decide is a `/dev` entry for the given interface. For the bus and PS/2 mice, there is little choice: the bus mouse always use `/dev/mse0`, and the PS/2 mouse is always at `/dev/psm0`. There may be more than one serial port to which the serial mouse can be attached. Many people often assign the first, built-in serial port `/dev/cuaa0` to the mouse.

If you are not sure which serial device your mouse is plugged into, the easiest way to find out the device is to use ``cat'' or ``kermit'' to look at the output of the mouse. Connect to it and just make sure that it generates output when the mouse is moved or clicked:

```
% cat < /dev/tty00
```

If you can't find the right mouse device then use ``dmesg|grep sio'' to get a list of serial devices that were detected upon booting:

```
% dmesg|grep sio
sio0 at 0x3f8-0x3ff irq 4 on isa
```

Then double check the `/dev` entries corresponding to these devices. Use the script `/dev/MAKEDEV` to create entries if they don't already exist:

```
% cd /dev
% sh MAKEDEV tty00
```

You may want to create a symbolic link `/dev/mouse` pointing to the real port to which the mouse is connected, so that you can easily distinguish which is your ``mouse'' port later.

The next step is to guess the appropriate protocol type for the mouse. In FreeBSD 2.2.6 or later, the X server may be able to automatically determine the appropriate protocol type, unless your mouse is of a relatively old model. Use the ``Auto'' protocol in these versions.

In other versions of FreeBSD or if the ``Auto'' protocol doesn't work in 2.2.6, you have to guess a protocol type and try.

There is rule of thumb:

1. The bus mice always use the ``BusMouse'' protocol regardless of the brand of the mouse.
2. The ``PS/2'' protocol should always be specified for the PS/2 mouse regardless of the brand of the mouse.

   NOTE: There are quite a few PS/2 mouse protocols listed in the man page for `XF86Config`. But, ``PS/2'' is the only PS/2 mouse protocol type useful in `XF86Config` for FreeBSD. The other PS/2 mouse protocol types are not supported in FreeBSD. FreeBSD version 2.2.6 and later directly support these protocol types in the PS/2 mouse driver `psm` and it is not necessary to tell the X server which PS/2 mouse protocol type is to be used; ``Auto'' should work, otherwise use ``PS/2''.
3. The ``Logitech'' protocol is for old mouse models from Logitech. Modern Logitech mice use either the ``MouseMan'' or ``Microsoft'' protocol.
4. Most 2-button serial mice support the ``Microsoft'' protocol.
5. 3-button serial mice may work with the ``MouseSystems'' protocol. If it doesn't, it may work with the ``Microsoft'' protocol although the third (middle) button won't function. 3-button serial mice may also work with the ``MouseMan'' protocol under which the third button may function as expected.
6. 3-button serial mice may have a small switch to choose between ``MS'' and ``PC'', or ``2'' and ``3''. ``MS'' or ``2'' usually mean the ``Microsoft'' protocol. ``PC'' or ``3'' will choose the ``MouseSystems'' protocol.
7. If the serial mouse has a roller or a wheel, it may be compatible with the ``IntelliMouse'' protocol.

5. Running X

8mb of memory is a recommended minimum for running X. The server, window manager, display manager and an xterm take about 8Mb of virtual memory themselves. Even if their resident set size is smaller, on a 8Mb system that leaves very space for other applications such as gcc that expect a few meg free. The R6 X servers may work with 4Mb of memory, but in practice compilation while running X can take 5 or 10 times as long due to constant paging.

The easiest way for new users to start X windows is to type ``startx >& startx.log''. Error messages are lost unless you redirect them because the server takes over the screen.

To get out of X windows, type: ``exit'' in the console xterm. You can customize your X by creating `.xinitrc`, `.xserverrc`, and `.twmrc` files in your home directory as described in the *xinit* and *startx* man pages.

6. Rebuilding Kernels for X

The GENERIC FreeBSD kernels support XFree86 without any modifications required. You do not need to make any changes to the GENERIC kernel or any kernel configuration which is a superset.

For a general description of BSD kernel configuration get smm.02.config.ps.Z. It is a ready-to-print postscript copy of the kernel configuration chapter from the system maintainers manual.

If you do decide to reduce your kernel configuration file, do not remove the line below (in `/sys/arch/i386/conf`). It is required for X support:

```
options                 UCONSOLE                #X Console support
```

The generic FreeBSD kernels are configured by default with the syscons driver. To configure your kernel similarly it should have a line like this in `/usr/src/sys/i386/conf/GENERIC`:

```
device          sc0     at isa? port "IO_KBD" tty irq 1 vector scintr
```

The number of virtual consoles can be set using the MAXCONS option:

```
options                 "MAXCONS=4"             #4 virtual consoles
```

Otherwise, the default without a line like this is 16. You must have more VTs than gettys as described in the end of section 3, and 4 is a reasonable minimum.

The server supports two console drivers: syscons and pcvt. The syscons driver is the default in FreeBSD 1.1.5 and higher. They are detected at runtime and no configuration of the server itself is required.

If you intend to use pcvt as the console driver, be sure to include the following option in your kernel configuration file.

```
options                 XSERVER                 #Xserver
```

The number of virtual consoles in pcvt can be set using the following option:

```
options                 "PCVT_NSCREENS=10"      #10 virtual consoles
```

The bus mouse driver and the PS/2 mouse driver may not be included, or may be included but disabled in your kernel. If you intend to use these mice, verify the following lines in the kernel configuration file:

```
device          mse0    at isa? port 0x23c tty irq 5 vector mseintr
device          psm0    at isa? port "IO_KBD" conflicts tty irq 12 vector psmintr
```

The `mse0` device is for the bus mouse and the `psm` device is for the PS/2 mouse. Your bus mouse interface card may allow you to change IRQ and the port address. Please refer to the manual of the bus mouse and the manual page for `mse(4)` for details. There is no provision to change IRQ and the port address of the PS/2 mouse.

The XFree86 servers include support for the MIT-SHM extension. The GENERIC kernel does not support this, so if you want to make use of this, you will need a kernel configured with SYSV shared memory support. To do this, add the following line to your kernel config file:

```
options                 SYSVSHM                 # System V shared memory
options                 SYSVSEM                 # System V semaphores
```

```
options         SYSVMSG         # System V message queues
```

If you are using a SoundBlaster 16 on IRQ 2 (9), then you need a patch for sb16_dsp.c. Otherwise a kernel configured with the SoundBlaster driver will claim interrupt 9 doesn't exist and X server will lock up.

S3 cards and serial port COM 4 cannot be installed together on a system because the I/O port addresses overlap.

README for XFree86 on FreeBSD **:** *Rebuilding Kernels for X*
Previous: *Running X*
Next: *Building X Clients*

7. Building X Clients

The easiest way to build a new client (X application) is to use `xmkmf` if an `Imakefile` is included with it. Type ``xmkmf -a'' to create the Makefiles, then type ``make''. Whenever you install additional man pages you should update `whatis.db` by running ``makewhatis /usr/X11R6/man''.

Note: Starting with XFree86 2.1 and FreeBSD 1.1, the symbol **__386BSD__** no longer gets defined either by the compiler or via the X config files for FreeBSD systems. When porting clients to BSD systems, make use of the symbol **BSD** for code which is truly BSD-specific. The value of the symbol can be used to distinguish different BSD releases. For example, code specific to the Net-2 and later releases can use:

```
#if (BSD >= 199103)
```

To ensure that this symbol is correctly defined, include `<sys/param.h>` in the source that requires it. Note that the symbol **CSRG_BASED** is defined for *BSD systems in XFree86 3.1.1 and later. This should be used to protect the inclusion of `<sys/param.h>`.

For code that really is specific to a particular i386 BSD port, use **__FreeBSD__** for FreeBSD, **__NetBSD__** for NetBSD, **__OpenBSD__** for OpenBSD, **__386BSD__** for 386BSD, and **__bsdi__** for BSD/386.

8. Thanks

Many thanks to:

- **Pace Willison** for providing initial *BSD support.
- **Amancio Hasty** for 386BSD kernel and S3 chipset support.
- **David Greenman, Nate Williams, Jordan Hubbard** for FreeBSD kernel support.
- **Rod Grimes**, **Jordan Hubbard** and **Jack Velte** for the use of Walnut Creek Cdrom's hardware.
- **Orest Zborowski**, **Simon Cooper** and **Dirk Hohndel** for ideas from the Linux distribution.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/FreeBSD.sgml,v 3.25.2.5 1998/11/07
13:37:46 dawes Exp $
```

```
$XConsortium: FreeBSD.sgml /main/12 1996/10/28 05:43:08 kaleb $
```

Information for ISC Users

Michael Rohleder

11 January 1998

1. X11R6/XFree86[tm] on Interactive Unix
2. Things needed for compiling the sources
3. Changes to the System Header Files
    3.1. /usr/include/sys/limits.h
    3.2. /usr/include/sys/ioctl.h
    3.3. /usr/include/errno.h
    3.4. /usr/include/rpc/types.h
    3.5. /usr/include/sys/un.h
    3.6. /usr/include/math.h
4. make World
5. linear Addressing
6. XKeyboard Extension
7. Multibuffer Extension
8. Sample Definitions
9. Installation
10. Using ...
11. Acknowledgements

1. X11R6/XFree86[tm] on Interactive Unix

This document provides some additional information about compiling and using X11R6 and XFree86 on your Interactive Unix, also referred to as ISC.

If you have any suggestions, comments, fixes or ideas regarding X11R6/XFree86 on Interactive Unix, send e-mail to

michael.rohleder@stadt-frankfurt.de

Bug Reports should be sent to

XFree86@XFree86.Org

Questions or anything else should be posted to the NewsGroup

comp.windows.x.i386unix

There is currently no support for shared Libraries so it will be filespace consuming if you want to build X11-clients with X11R6. Best you mix X11R6 Server with X11R5 and X11R4 clients. And only compile clients who need the new facilities provided in the X11R6 Libraries against them.

2. Things needed for compiling the sources

gcc

Use the highest number for x you found. Fresco will only build 2.6.3 and later. I'd tried gcc Version 2.5.8, 2.6.0, 2.6.2, 2.6.3 and 2.7.2. Current: 2.7.2.3

Since 2.6.3 the current source tree should be able to compile with a little bit more Optimization: `#define DefaultCDebugFlags -O3 -fomit-frame-pointer` inside xf86site.def to overwrite the default -O2.

With 2.7.x you must specify `#define UsePosix YES` inside xf86site.def. This is necessary to build the sources successfully. Versions prior to 2.7.0 could define it, but don't need it for a clean build.

libg++-2.x.x

The needed g++ Libraries for use with g++ 2.x.x. As this is only necessary for Fresco, it isn't needed anymore since X11R6.1.

binutils

You could use the assembler and linker the assembler is most preferred,and the linker is needed at least if you want to link libFresco.a within a Program. Don't use strip and ar/ranlib, the first generates buggy binaries when stripping (at least on my machines) and the last requires the use of ranlib after creating an archive, this is not configured. Current: 2.8.1.0.15 (Used: as, ld, ar, strip)

gnu-malloc

Due to better memory usage we should use GNU's malloc library on systems where possible.

Enable `#define UseGnuMalloc YES` inside xf86site.def or within the Linkkit site.def.

Enable and set `#define GnuMallocLibrary` to your needs, if it isn't like the default `-L/usr/local/lib -lgmalloc`.

inline-math (optional)

This is the "original" inline-math package available at your favorite Linux Mirror.

Use `#define UseInlineMath YES` inside host.def to enable it. Please note the changes section what else to do, to use this package.

3. Changes to the System Header Files

You have to change some of the standard header files supplied with your version of Interactive. You also need to change some of the include files in the gcc-lib/include directory.

Let us say the gcc-files are in directory

`/usr/local/lib/gcc-lib/i[345]86-isc[34].[0-9]/2.6.x`

referred to as `"gcc-lib"`

3.1. `/usr/include/sys/limits.h`

and `gcc-lib/include/sys/limits.h`

```
#ifndef OPEN_MAX
#ifdef ISC
#define OPEN_MAX        256
#else
#define OPEN_MAX        20
#endif
#endif
```

`OPEN_MAX` had to be increased to prevent Xlib Errors (max no. of clients reached).

3.2. `/usr/include/sys/ioctl.h`

surrounded by

```
#ifndef _IOCTL_H
#define _IOCTL_H
...
#endif
```

to prevent multiple includes.

3.3. `/usr/include/errno.h`

(and the corresponding gcc-include-file) add

```
#include <net/errno.h>
```

because of **EWOULDBLOCK** undefined in several places regarding lbx. Surround

`/usr/include/net/errno.h` with

```
#ifndef _NET_ERRNO_H
#define _NET_ERRNO_H
...
#endif
```

to prevent multiple includes were `<net/errno.h>` is explicit included from the sources.

3.4. `/usr/include/rpc/types.h`

copy this file to `gcc-lib/include/rpc/types.h` and change the declaration of **malloc()** to

```
#if !defined(__cplusplus)
extern char *malloc();
#endif
```

Note that this is only necessary if you want to build Fresco

3.5. `/usr/include/sys/un.h`

such a file does not exist on Interactive. You may like to generate it, if you don't like a warning from depend. It isn't needed to compile the sources successfully.

You could use the following to produce it:

```
#ifndef X_NO_SYS_UN
struct  sockaddr_un {
        short   sun_family;             /* AF_UNIX */
        char    sun_path[108];          /* path name (gag) */
};
#endif
```

3.6. `/usr/include/math.h`

To use the Inline Math package you have to change your existing math.h. Please note, the way I include the new Header file, is different than suggested in inline-math's README.

Please add the following at the bottom of math.h, before the last #endif

```
#if defined(UseInlineMath)

/* Needed on ISC __CONCAT, PI */
#ifndef __CONCAT
/*
 * The __CONCAT macro is used to concatenate parts of symbol names, e.g.
```


```
 * with "#define OLD(foo) __CONCAT(old,foo)", OLD(foo) produces oldfoo.
 * The __CONCAT macro is a bit tricky -- make sure you don't put spaces
 * in between its arguments.  __CONCAT can also concatenate double-quoted
 * strings produced by the __STRING macro, but this only works with ANSI C.
 */
#if defined(__STDC__) || defined(__cplusplus)
#define __CONCAT(x,y)   x ## y
#define __STRING(x)     #x
#else   /* !(__STDC__ || __cplusplus) */
#define __CONCAT(x,y)   x/**/y
#define __STRING(x)     "x"
#endif  /* !(__STDC__ || __cplusplus) */
#endif

#ifndef PI
#define PI M_PI
#endif

#include "/usr/local/include/i386/__math.h"
#endif
```

Information for ISC Users : *Changes to the System Header Files*
Previous: *Things needed for compiling the sources*
Next: *make World*

4. make World

```
BOOTSTRAPCFLAGS="-DISC [-DISC30 | -DISC40] -DSYSV [-Di386]"
```

-DISC -DISC30

these two defines are necessary to build the release I don't know if the build will succeed for ISC versions prior than 3.x

-DISC40

are only for getting the ISC version and therefore set the HasSymLinks to Yes ('cause symbolic linking were only supported from Version 4.x using the S5L Filesystem)

If you could use long filenames, you could enable the installation of expanded Manual Pages by including `#define ExpandManNames YES` inside xf86site.def.

A build on ISC 4.x only needs -DISC40 defined in the BOOTSTRAPCFLAGS (-DISC30 will be included automatically).

Note: due to some incompatibilities between ISC 4.0 and 4.1, the default is to build for ISC4.0, even if you build on 4.1. If you want to build only for 4.1 you should set `#define IscCompileVersion 410` inside your host.def.

(the fchmod function isn't available on 4.0, so it won't compile, and binaries from 4.1 won't run cause of the unsupported System call The libraries build for 4.1 couldn't be used with 4.0 Systems, due to some functions not available on 4.0)

-DSYSV [-Di386]

standard defines for SystemV Release3 on x86 platform. You don't need to explicitly define -Di386 because this is pre-defined in /lib/cpp.

---

5. linear Addressing

- Compiling ...

  If you want to include support for linear addressing into the server binaries, you have to define

  ```
  #define HasSVR3mmapDrv            YES
  ```

  in xf86site.def. This is necessary to get the correct setup to be defined for the build.

  You need the mmap-2.2.3 driver installed on your system. If you don't have the mmap-2.2.3 driver installed, you could use the driver source in the file

  ```
  xc/programs/Xserver/hw/xfree86/etc/mmapSVR3.shar
  ```

  or

  ```
  /usr/X11R6/lib/X11/etc/mmapSVR3.shar
  ```

  Build and install the driver as instructed. You'll need the file /usr/include/sys/mmap.h for compiling the X11R6/XFree86 source tree, with linear addressing enabled.
- Using ...

  To use the linear address-mapping of the framebuffer you need the mmap Driver by Thomas Wolfram (Version 2.2.3) installed in your Kernel. If you have installed it, most servers will use linear addressing by default. Others may require setting the

  ```
  Option "linear"
  ```

  in your XF86Config. Check the appropriate manual pages for details. Maybe you need also the MemBase specified in XF86Config. Please refer to the appropriate README of your Card/Server, for **How to use...** Note that the P9000 server will not work at all unless linear addressing is available.

  I could only test these cards against the linear addressing.

  - Spea/V7 Vega - clgd5428 - VLB

    with 32MB MainMemory installed I couldn't use it. My tests with different mappings into the address space results in no Graphics displayed or a spontaneous reboot.
  - ATI GUP - mach32 - VLB

    with 32MB MainMemory installed I could map the CardMemory at MemBase 0x07c00000. I could work with all clients until I try to activate a Motif 1.1.1 InputField inside a Motif

Client like Mosaic-2.4 or xplan. This results in a crash of the XServer.

```
!!! You could work around this !!!
```

Expand your .Xdefaults with

```
*blinkRate:                    0
*cursorPositionVisible:        false
```

This bug seems to be fixed since 3.1.2, and therefore the workaround is not needed anymore.

- ELSA Winner 2000PRO/X Revision G

  if you experience a Problem with this Card you could try to use the older Chipset Driver instead "newmmio".

  If you declare

  ```
  Chipset "mmio_928"
  ```

  inside your XF86Config, it may be alright again.

  With the current XF86_S3 I don't encounter any problem.

Information for ISC Users **:** *linear Addressing*
Previous: *make World*
Next: *XKeyboard Extension*

6. XKeyboard Extension

- Sample Setup ...

  Here is a sample XKeyboard Definition to include inside the Keyboard Section of your XF86Config File.

```
    Xkbkeycodes "xfree86"
/*    XkbSymbols  "us(pc101)+de_nodead"  */
/*    This has changed between 3.1.2E and 3.1.2F */
/*    it is now:                                 */
    XkbSymbols  "us(pc102)+de(nodeadkeys)"
    XkbTypes    "default"
    XkbCompat   "default"
    XkbGeometry "pc"
```

  or you could use this one with the new Options:

```
   XkbRules        "xfree86"
   XkbModel        "pc102"
   XkbLayout       "de"
   XkbVariant      "nodeadkeys"
```

TOC

7. Multibuffer Extension

This is an obsolete Extension. Anyway, if you want to include this Extension inside your build, you have to add: `#define BuildMultibuffer YES` inside xf86site.def Please note, this Extension should be disabled when building the Loader Server.

8. Sample Definitions

This is my current host.def, if I build the sources. (So no more changes were necessary in xf86site.def, either it isn't to bad to have a look inside it ;-)

```
#ifdef BeforeVendorCF

/* ISC 4.1Mu - build only for 4.1
#define IscCompileVersion       410
*/

/* Use inline Math from linux ;-) package inline-math-2.6.tar.gz */
/* should be available on your favorite linux ftp                */
# define UseInlineMath          YES

/* Use cbrt from liboptm.a (Interactive icc Compiler) */
/*
*/
# define HasCbrt                YES

/* Use GNUs MallocLibrary (and the Location for the Lib)             */
# define UseGnuMalloc           YES
# define GnuMallocLibrary       -L/usr/local/lib -lgnumalloc

/* Build Xvfb                                                        */
# define XVirtualFramebufferServer      YES

/* Use mmap Driver                                                   */
# define HasSVR3mmapDrv         YES

/* Expand Manual Pages (needs S5L)                                   */
# define ExpandManNames         YES

/* Has LinuxDoc (and the Location for LinuxDoc / only HTML and Text) */
# define HasLinuxDoc            YES
# define BuildLinuxDocHtml      YES
# define BuildAllDocs           YES
# define LinuxDocDir            /usr/local/lib/linuxdoc-sgml
```

TOC

```
/* Install Config's for xdm, xfs, and xinit */
# define InstallXinitConfig    YES
# define InstallXdmConfig      YES
# define InstallFSConfig               YES

#define BuildChooser            YES

/* for the new XF86Setup Util */
#define HasTk                    YES
#define HasTcl                   YES

#endif /* BeforeVendorCF */
```

Information for ISC Users **:** *Sample Definitions*
Previous: *Multibuffer Extension*
Next: *Installation*

9. Installation

After your **make World BOOTSTRAPCFLAGS="...** succeed,

`make install`

to install in /usr/X11R6. Make sure you have enough space, and /usr/X11R6 exists either as a directory or a symlink to another directory maybe in another filesystem.

`make install.man`

to install the compressed nroff versions of the manual pages into /usr/X11R6/man. This directory will be generated if it doesn't exist.

`make install.linkkit`

to install the server binary LinkKit into /usr/X11R6/lib/Server.

```
      You could tune the Kernel using the command-file

      /usr/X11R6/lib/X11/etc/xf86install

      This will increase the available pseudo devices,
      some Tunable Parameters and install some files
      to use inside sysadm. You could also install
      some additional Fonts and Terminal files.
```

You also should increase MAXUMEM to its maximum, else programs may die with:

```
X Error of failed request:  BadAlloc (insufficient resources for operation)
  Major opcode of failed request:  53 (X_CreatePixmap)
  Serial number of failed request:  37791
  Current serial number in output stream:  37822
  Widget hierarchy of resource: unknown
```

---

10. Using ...

- Xprt:

  The new Xprint Server is configured to use lpr as its print helper so you have to install and configure lpr to use Xprt.
- Keyboard:

  You don't need any modmap-File to get your keyboard working with any iso-8859-1 Font. Simply enable
  - LeftAlt Meta
  - RightAlt ModeShift
  - RightCtl Compose

  in your XF86Config - Section "Keyboard"
- xpcterm:

  if you want to get the German 'Umlaut' inside your ISC X11R4 client xpcterm when you are using the ega/vga font. Set up the user's .Xdefaults to contain:

```
XEga*AT386.Translations: #override \
          Shift<Key>odiaeresis: string(0x99) \n\
               <Key>odiaeresis: string(0x94) \n\
          Shift<Key>adiaeresis: string(0x8e) \n\
               <Key>adiaeresis: string(0x84) \n\
          Shift<Key>udiaeresis: string(0x9a) \n\
               <Key>udiaeresis: string(0x81) \n\
          Shift<Key>ssharp: string(0x3f) \n\
           Meta<Key>ssharp: string(0x5c)  \n\
               <Key>ssharp: string(0xe1)
```

  The only disadvantage is that you have to use `Alt` instead of `AltGr` to get the `\` Backslash (on a German Keyboard)

  You have to call your xpcterm with the option `-name XEga -fn ega`
- Switching between X11R5 and X11R6 configuration

  to compile X11-Clients as either R6 or R5 clients, should be as easy as you only switch the PATH components so that either `/usr/X11R6/bin/xmkmf` or `/usr/X386/bin/xmkmf` would make the new Makefile.
- ISC Streams Pipes

  The old path to the pipes on ISC's R4 `/tmp/.X11-unix` has changed to `/dev/X/ISCCONN`. For compatibility reasons on ISC, the pipes in the new directory will be linked to a file inside the old. This will normally be a hard link, so it can't go across filesystems. On ISC Version 4.x this is now allowed. But you should use the new S5L on both filesystems. ISC30 systems should take care that the two directories are on the same FS. Else if you are using a ISC40 compiled binary, the Server could maybe abort due to a SIGSYS. We tried to catch this signal, so if it dumps please send me a note.
- Warnings you may see:
  - Since 3.2A, you could see a warning from pre X11R6.3 clients.

```
Warning: Unable to load any usable fontset
```

    The case are the new gzipped fonts, but the Warning isn't serious.
  - If you start a server you may see the following message:

```
_XSERVTransOpen: transport open failed for named/enigma:0
_XSERVTransMakeAllCOTSServerListeners: failed to open listener for n
```

TOC

This message either isn't critical. Interactive doesn't support this kind of connection.

Information for ISC Users **:** *Using ...*
Previous: *Installation*
Next: *Acknowledgements*

11. Acknowledgements

All thanks should go to the members of the **XFree86 Team** for their great work and the **X Consortium** for their Public Release of X11R6, as to all who contribute to this excellent piece of free software.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/isc.sgml,v 3.18.2.2 1998/02/20
23:10:31 dawes Exp $
```

```
$XConsortium: isc.sgml /main/8 1996/10/23 11:45:58 kaleb $
```

Information for Linux Users

Orest Zborowski, Dirk Hohndel

June 25, 1999

1. Linux versions on which XFree86 has been tested
2. Backwards Compatibility
3. Installing XFree86
4. Running XFree86
5. Installing Xdm, the display manager

1. Linux versions on which XFree86 has been tested

XFree86 has been tested with Linux version 2.0.36, 2.2.7 and several 2.3.x kernels. It should work with any version since 1.0 without change. Binaries both against libc5 and libc6 are available.

2. Backwards Compatibility

X11R6 is considered a major update from X11R5, so the shared libraries in XFree86 3.1 and later are not compatible with XFree86 2.1.1 and older libraries. To continue to run X11R5 applications, you must keep the old libraries somewhere on your machine. They can be moved from `/usr/X386/lib` elsewhere, but `/etc/ld.so.conf` must be updated. All X11R5 applications should work with the X11R6 servers without problems.

X11R6.1 is yet another update to X11R6. While the minor number for some libraries has been increased to '1' it is believed to be fully compatible with X11R6 based applications.

X11R6.3 is yet another update to X11R6.1. While the minor number for some libraries has been increased to '3' it is believed to be fully compatible with X11R6 based applications.

Very old binaries (linked to XFree86-1.2, XFree86-1.3 or XFree86-2.0 libraries) will continue to work, but may need an explicit symlink from `/lib/libX{11,t,aw}.so.3` to `/usr/X386/lib/libX{11,t,aw}.so.3`.

3. Installing XFree86

Starting with version 3.0, XFree86 is installed in `/usr/X11R6`. The installation details are provided in the RELNOTES.

4. Running XFree86

XFree86 requires about 4mb of virtual memory to run, although having 16mb of RAM is probably the minimum comfortable configuration. A 387 coprocessor is helpful for 386 machines, although greater gains in interactive performance are obtained with an increase in physical memory. Also, a faster graphics card, bus or RAM, will improve server performance.

After unpacking the tar files, you need to include `/usr/X11R6/lib` in `/etc/ld.so.conf` (where it should already be by default) or in your `LD_LIBRARY_PATH` environment variable. Also, the configuration file `/etc/XF86Config` or `/usr/X11R6/lib/X11/XF86Config` *must* be properly filled out based on the host setup. Ideally this is done using `XF86Setup` or (if for some reason this doesn't work) using `xf86config`. If you really insist in hand-creating your config file use `XF86Config.eg` as a starting point and `README.Config` as guideline. You may damage your hardware if you use a wrong `XF86Config` file, so *read the docs*, especially the man pages and the other `README` files in `/usr/X11R6/lib/X11/doc`.

XFree86 has the ability to perform VT switching to and from the X server. When first started, XFree86 will automatically locate the first available VT (one that hasn't been opened by any process), and run on that VT. If there isn't one available, XFree86 will terminate with an error message. The server can be run on a specific VT by using the ``vt<nn>'' option, where <nn> is the number of an available VT (starting from 1). If you don't have a free VT XFree86 cannot run. Normally you can simply disable one of the `getty` programs in `/etc/inittab`, but if this is not an option, you can increase the number of available VTs by increasing the value of `NR_CONSOLES` in `include/linux/tty.h` and recompiling the kernel.

Once running inside X, switching to another VT is accomplished by pressing Ctrl-Alt-<Fnn> where nn is the number of the VT to switch to. To return to the server, press the proper key-combination that moves you back to the VT that XFree86 is using: by default, this is Alt-<Fmm>, where mm is the number of the VT the server is running on (this number is printed when the server is started). Note that this is NOT the VT from which the server was started.

NOTE: you can redefine the text-mode keybindings with the `` `loadkeys' `` command found in the `kbd-0.81.tar.gz` archive (or a later version thereof). With this, you can (for example) make Ctrl-Alt-<Fmm> work from text mode the same way it works under the XFree86 server.

When the server is exited, it will return to the original VT it was started from, unless it dies unexpectedly, when the switch must be done manually. There still seem to be weird combinations of graphic cards and motherboards that have problems to restore the textfont when returning from XFree86 to the text mode. In these cases using the `runx` script from the **svgalib** distribution might help.

The XFree86 server now queries the kernel to obtain the key binding in effect at startup. These bindings are either the default map in place when the kernel was compiled, or reloaded using the `` `loadkeys' ``

utility. Not all keys are bound: kernel-specific, multiple keysym, and dead keys are not handled by the server. All others are translated to their X equivalents. Note that the XFree86 server only allows for four modifier maps: unshifted, shifted, modeswitch unshifted and modeswitch shifted. Depending on what the modeswitch key is (it is configurable in your `XF86Config` and defaults to Alt), XFree86 will read those tables into its keymaps. This means if you use certain keys, like left-Control, for Linux modeswitch, that will not be mappable to X.

Information for Linux Users **:** *Running XFree86*
Previous: *Installing XFree86*
Next: *Installing Xdm, the display manager*

5. Installing Xdm, the display manager

Since xdm is dynamically linked, there's no issue on export restriction outside US for this binary distribution of xdm: it does not contain the DES encryption code. So it's now included in the bin package.

However the file `xc/lib/Xdmcp/WrapHelp.c` is not included in the XFree86-3.3 source, so support for XDM-AUTHORIZATION-1 is not included here. You'll have to get WrapHelp.c and rebuild xdm after having set `HasXdmAuth` in `xf86site.def`.

The file is available within the US; for details see ftp.x.org:/pub/R6/xdm-auth/README.

To start the display manager, log in as root on the console and type: ``xdm -nodaemon''.

You can start xdm automatically on bootup by disabling the console getty and modifying `/etc/inittab`. Details about this setup depend on the Linux distribution that you use, so check the documentation provided there.

The xdm binary provided should run with both shadow- and non-shadow password systems.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Linux.sgml,v 3.13.2.8 1999/06/25
08:57:14 hohndel Exp $
```

```
$XConsortium: Linux.sgml /main/6 1996/10/28 04:47:37 kaleb $
```

README for XFree86 on LynxOS

TOC

Thomas Mueller

Last modified on: 30 May 1999

1. What and Where is XFree86?
2. Installing the Binaries
3. Running XFree86
   - 3.1. System requirements
   - 3.2. System tuning
   - 3.3. Mouse support in XFree86
   - 3.4. Bus mouse drivers
   - 3.5. ATC console driver and VT switching
   - 3.6. X Server debug diagnostics output and other VT peculiarities
4. Installing XFree86 manual pages
5. Using XFree86 with Motif
   - 5.1. Copy Motif files
   - 5.2. Motif library patch for LynxOS AT 2.3.0
   - 5.3. X11R6 config file patch
   - 5.4. Motif config file patch
6. Compiling the XFree86 Distribution
   - 6.1. Disk space requirements
   - 6.2. Changes to system environment (LynxOS AT)
   - 6.3. make World
7. Building on microSPARC and PowerPC
   - 7.1. Console driver patch for microSPARC
   - 7.2. Known Bug of the microSPARC server

TOC

1. What and Where is XFree86?

XFree86 is a port of X11R6.3 that supports several versions of Intel-based Unix. It is derived from X386 1.2, which was the X server distributed with X11R5. This release consists of many new features and performance improvements as well as many bug fixes. The release is available as source patches against the X Consortium X11R6.3 code, as well as binary distributions for many architectures.

See the Copyright Notice in Copyright Notice.

The sources for XFree86 are available by anonymous ftp from:

ftp://ftp.XFree86.org/pub/XFree86/current

Binaries of XFree86 for LynxOS AT are available from:

ftp://ftp.XFree86.org/pub/XFree86/current/binaries/LynxOS

The binaries were built on `LynxOS x86 3.0.1'. Because of changes made to the object format they don't run on LynxOS versions earlier than 3.0.0.

Building of this XFree86 version has never been tested on LynxOS versions earlier than 2.4.0. Binaries built on LynxOS 2.4.0 are expected to run on 2.3.0 as well.

XFree86 supports LynxOS on the AT, on the microSPARC and on the PowerPC platform. X servers are currently available on the AT and microSPARC platform. Refer to section Building on microSPARC and PowerPC for details on XFree86 on the non-AT platforms.

If you need binaries for other platforms than the one on the XFree86 FTP server contact me (tmueller@sysgo.de).

Send email to *tmueller@sysgo.de* (Thomas Mueller) or *XFree86@XFree86.org* if you have comments or suggestions about this file and we'll revise it.

2. Installing the Binaries

Please refer to section "Installing the XFree86 3.3.4 Release" of the Release Notes for detailed installation instructions.

If you plan to install XF86Setup you'll have to install `X333prog` as well since XF86Setup checks for the existence of a certain file name pattern which is satisfied only if you install the library files from `X333prog`.

It may be necessary to increase the process stack limit in order to run XFree86 on your system. Edit `/etc/startab` and reboot your system to make the changes active before you begin the installation.

Also, be sure to include `/usr/X11R6/bin` in your PATH environment variable.

Refer to the next section Running XFree86 for further information on necessary configuration steps before running XFree86 on LynxOS.

3. Running XFree86

This section describes the changes to the LynxOS environment which may be necessary to successfully run XFree86.

Read Quick-Start Guide to XFree86 Setup to learn more about how to configure XFree86 for your hardware.

3.1. System requirements

A minimum of 16MB of memory is required to run X. If you want to run real-world applications you should think of upgrading to 32MB (or more).

3.2. System tuning

3.2.1. Tunable parameters

To reasonably run XFree86 you may have to adjust a few system parameters.

On LynxOS 2.5.x and 3.0.x include a line

```
#define X_WINDOWS
```

in `/sys/lynx.os/uparam.h`.

For earlier versions you'll have to edit `/usr/include/param.h`:

```
Tunable                                          Old     New
USR_NFDS      number of open files per process    20      64
NPROC         number of tasks                     50     150
NFILES        number of open files in system     100     250
NINODES       number of incore inodes            (same value as NFILES)
QUANTUM       clock ticks until preemption        64      20
CACHEBLKS     number of cache memory blocks      202      >= 4096
```

The new values are those suggested by the LynxOS documentation for their X Window package.

3.2.2. Adjustment for Riva 128 and Riva TNT driver>If you're using the nVidia driver (Riva 128, TNT, TNT2) of the SVGA server, you will have to increase the value of the SMEMS parameter in `/sys/lynx.os/uparam.h` from 10 to 20.

3.2.3. Increase number of ptys

You should also increase the number of ptys to be able run a couple more xterms. You may replace `/sys/lynx.os/pty.cfg` with `/usr/X11R6/lib/X11/etc/pty.cfg`.

3.2.4. Kernel build

If you plan to use PS/2 or Bus mice refer to the following section before rebuilding the kernel, if not, you should rebuild the kernel now:

```
# cd /sys/lynx.os
# make install
# reboot -N
```

3.3. Mouse support in XFree86

XFree86 includes support for PnP mice (see also Mouse Support in XFree86). The current LynxOS TTY device driver doesn't allow the necessary manipulation of the RTS line and therefore the support for PnP mice has been disabled for LynxOS.

3.4. Bus mouse drivers

Starting with LynxOS AT 2.4.0 LynxOS includes a PS/2 mouse driver. Currently this driver is not fully supported by XFree86 (you'll probably have to specify the mouse type as *Microsoft* regardless of real mouse type and in some cases you won't have all mouse buttons supported).
`/usr/X11R6/lib/X11/etc/BM-Lynx.shar` contains a LynxOS port of the Linux bus mouse drivers. To install the drivers unpack the shar archive

```
# cd /
# bash /usr/X11R6/lib/X11/etc/BM-Lynx.shar
```

and follow the notes in `/BMOUSE.Readme` for further installation and configuration notes.

The XFree86 PS/2 mouse driver works also with MetroLink X 2.3.3.1 as shipped with LynxOS AT 2.4.0 unless you have the LynxOS patch 000055-00 installed.

3.5. ATC console driver and VT switching

The XFree86 servers will only run with the default LynxOS console driver, sorry for those of you who use the alternative vdt console driver. Currently there is no support for virtual terminal switching once the server has started.

You will need a free console which the X server will use for keyboard input. You must disable login on at least one of the four virtual terminals in `/etc/ttys`, e.g. `/dev/atc3`:

```
change

        /dev/atc3:1:default:vt100at:/bin/login

to
```

```
/dev/atc3:0:default:vt100at:/bin/login
          ^
```

TOC

3.6. X Server debug diagnostics output and other VT peculiarities

The XFree86 X servers will produce a lot of diagnostics output on stderr during startup. This output will be lost after the server reached a certain point in its console initialization process. You should redirect stdout and stderr if you want to analyze the diagnostics produced by the server.

When the X server is running output made to other consoles will be lost. After server shutdown the screen contents of other consoles may be inconsistent with what one would expect (i.e. random).

---

4. Installing XFree86 manual pages

LynxOS uses cat-able manual pages, and because a doc preparation system is definitely not a vital component of a real-time operating system you must first install groff-1.09 (or newer). Starting with LynxOS 2.3.0 it should compile right out of the box (or better tar archive).

XFree86 manual pages may be installed using

```
make install.man
```

The index and whatis database for the XFree86 manual pages will be created automatically. If you already have a whatis database or index file in the destination directories you should perform a sort/uniq operation to remove duplicate entries:

```
for i in 1 3 5
do
  rm -f /tmp/tmpfile
  sort /usr/X11R6/man/cat$i/LIST$i | uniq > /tmp/tmpfile
  mv /tmp/tmpfile /usr/X11R6/man/cat$i/LIST$i
done
sort /usr/X11R6/man/whatis | uniq > /tmp/tmpfile
mv /tmp/tmpfile /usr/X11R6/man/whatis
```

With LynxOS 2.3.0 you should include `/usr/X11R6/man` in the MANPATH environment variable.

```
bash: MANPATH=$MANPATH:/usr/X11R6/man
```

The man command of LynxOS 2.2.1 does not support the MANPATH environment variable properly. The XFree86 manual pages must be copied (or linked) to the standard manual page locations (`/usr/man/catx`) in order to be read the man command:

```
for i in 1 3 5
do
  ln -s /usr/X11R6/man/cat$i/*.* /usr/man/cat$i
  cat /usr/X11R6/man/cat$i/LIST$i >> /usr/man/cat$i/LIST$i
  sort -o /usr/man/cat$i/LIST$i /usr/man/cat$i/LIST$i
  cat /usr/X11R6/man/cat$i/whatis$i >> /usr/man/whatis
done
```

5. Using XFree86 with Motif

The Motif libraries shipped with LynxOS AT 2.3.0 and 2.4.0 can be used with the XFree86 libraries. Follow the steps outlined below after you have installed XFree86 and LynxOS Motif on your system.

5.1. Copy Motif files

You must create symbolic links for the Motif libraries and utilities in the `/usr/X11R6` directory tree.

```
ln -s /usr/bin/X11/uil /usr/X11R6/bin
ln -s /usr/lib/libUil.a /usr/X11R6/lib
ln -s /usr/lib/libMrm.a /usr/X11R6/lib
ln -s /usr/lib/libXm.a /usr/X11R6/lib
ln -s /usr/lib/X11/uid /usr/X11R6/lib/X11
ln -s /usr/include/Xm /usr/X11R6/include
ln -s /usr/include/Mrm /usr/X11R6/include
ln -s /usr/include/uil /usr/X11R6/include
```

The Motif imake-configuration files are part of the LynxOS X Window package. They must be copied to the `/usr/X11R6` directory tree.

```
cp /usr/lib/X11/config/Motif.* /usr/X11R6/lib/X11/config
```

5.2. Motif library patch for LynxOS AT 2.3.0

The XFree86 libraries are compiled with the -mposix compiler option while the Motif libraries shipped with LynxOS AT 2.3.0 are not. This incompatibility will cause Motif `XmFileSelection` widgets to be linked with the wrong (i.e. POSIX) directory routines. To circumvent this problem apply the following patch to the library:

```
cp /usr/lib/libXm.a /usr/X11R6/lib
ar x /usr/X11R6/lib/libXm.a Xmos.o
ar x /lib/libc.a directory.s.o
ld -r -o x.o Xmos.o directory.s.o
mv x.o Xmos.o
ar r /usr/X11R6/lib/libXm.a Xmos.o
```

This patch is not necessary for LynxOS revisions after 2.3.0.

5.3. X11R6 config file patch

Edit `/usr/X11R6/lib/X11/config/lynx.cf` and change the definition of `HasMotif`

from

```
#define HasMotif    NO
```

to

```
#define HasMotif    YES
```

5.4. Motif config file patch

The file `Motif.tmpl` shipped with LynxOS Motif must be modified to work with XFree86. In every reference to `UnsharedLibReferences` the first argument must be changed

from

```
UnsharedLibReferences(<Something>LIB, Arg2, Arg3)
```

to

```
UnsharedLibReferences(<Something>, Arg2, Arg3)
```

Be sure to apply the change to the file copied to `/usr/X11R6/lib/X11/config`.

README for XFree86 on LynxOS : *Using XFree86 with Motif*
Previous: *Installing XFree86 manual pages*
Next: *Compiling the XFree86 Distribution*

6. Compiling the XFree86 Distribution

Before trying to rebuild XFree86 from source read Building XFree86 for a detailed description of the build process. The next sections contain LynxOS specific notes with respect to the build process.

6.1. Disk space requirements

Currently there is no support for shared libraries in the LynxOS XFree86 port. A complete binary installation along with manual pages will require approximately 90-100 MBytes of disk space. To compile the system you will need at least 230 MBytes of free disk space.

6.2. Changes to system environment (LynxOS AT)

Before compiling the XFree86 distribution you will have to make a few little adjustments to your system:

LynxOS AT 2.5

- Create a shell script named `/lib/cpp` as follows:

```
#!/bin/sh
/usr/lib/gcc-lib/i386-unknown-lynxos2.5/2.7-96q1/cpp \
          -traditional "$@"
```

  On other platforms than the AT the paths for the compiler support programs are different. You may use

```
gcc -v
```

  to find out the correct path. Set the file mode of `/lib/cpp` with

```
# chown root /lib/cpp
# chmod 755 /lib/cpp
```

- Modify `/lib/liblynx.a`. The X servers need the `smem_create()` system call to map the frame buffer into their address space. The system call is in `liblynx` library along with other Lynx proprietary calls which (unfortunately) overlap with calls in `libc`. To reduce confusion you should modify `liblynx` as follows:

```
# mv /lib/liblynx.a /lib/liblynx.a.ORG
# mkdir /tmp/xx; cd /tmp/xx
# ar xv /lib/liblynx.a.ORG
# ar rv /lib/liblynx.a *smem*
# ranlib /lib/liblynx.a
```

LynxOS AT 2.4

- Use the CYGNUS GNU-C Compiler to build XFree86. With LynxOS 2.4.0 you must execute the shell script `/CYGNUS.bash` to apply the necessary changes to your environment.
- Create a shell script named `/lib/cpp` as follows:

```
#!/bin/sh
/cygnus/94q4-lynxos-x86/lib/gcc-lib/i386-lynx/2.6-94q4/cpp \
          -traditional "$@"
```

It is possible that future releases use a different path for the CYGNUS compiler support programs. You may use

```
gcc -v
```

to find out the correct path. Set the file mode of `/lib/cpp` with

```
# chown root /lib/cpp
# chmod 755 /lib/cpp
```

LynxOS AT 2.3

This has actually not been tested, but the steps for described for 2.4 should apply to 2.3 as well.

LynxOS AT 2.2.1

This has actually never been tested, be prepared that the build will fail somewhere!

- Create a shell script named `/lib/cpp` as follows:

```
#!/bin/sh
/usr/local/lib/gcc-cpp -traditional "$@"
```

- The loader /bin/ld of LynxOS 2.2.1 does not support the -L option which is heavily used by X11R6 makefiles. To work around this problem you must install a small wrapper program which replaces the original /bin/ld program. Use the following steps to install it:

```
# cd xc/programs/Xserver/hw/xfree/etc
# cc -o ld ld-wrapper.c
# mv /bin/ld /bin/ld.org
# mv ld /bin/ld
# chmod 511 /bin/ld
# chown root /bin/ld
```

- Modify system header files as follows:

`/usr/include/uio.h`

surrounded by

```
#ifndef _UIO_H
```


```
#define _UIO_H
...
#endif
```

/usr/include/utmp.h

surrounded by

```
#ifndef _UTMP_H
#define _UTMP_H
...
#endif
```

/usr/include/unistd.h

add

```
extern int read();
```

6.3. make World

Read Building XFree86 before trying to rebuild XFree86 from the source distribution.

You may then issue a

```
make World
```

to compile XFree86. After a few hours (and hopefully a successful build of the XFree86 system) you can install the software using

```
make install
```

You must be logged in as super-user (root) when you invoke `make install'. Be sure to set your environment to use the same compiler (LynxOS 2.3.0/2.4.0, CYGNUS GNU-C) as you did during the `make World'. To install the LinkKit use

```
make install.linkkit
```

With LynxOS 2.2.1 programs will not be stripped during installation. This is due to a problem with the strip program which shows up when installing across file system boundaries.

Refer to section Installing XFree86 manual pages for manual page installation.

On LynxOS AT 2.5.0 you may encounter problems with `make` in deeply nested subdirectories (eg core dumps, hangups). In this case update to GNU make version 3.75 or higher.

7. Building on microSPARC and PowerPC

XFree86 3.3 compiles on LynxOS microSPARC and on LynxOS PPC as well. On the microSPARC there is X server support for the colour frame buffers CG3 and CG6 while on the PPC there is no X server available at this time. Before you start the build (on versions earlier than 2.5.0) you must create a symbolic link from the CYGNUS gcc to a file named `cc` somewhere in a directory included in your PATH environment variable.

7.1. Console driver patch for microSPARC

Before building on the microSPARC you should install the patch for the console driver supplied in `xc/programs/Xserver/hw/sunLynx/patch.Console`. (`xc/programs/Xserver/hw/sunLynx/patch.Console-2.4.0` for LynxOS revisions earlier than 2.5.0). The patch fixes minor problems in the original LynxOS driver and adds functionalities to detect the keyboard type and control the key click. To create a backup of the original driver and install the patch issue the commands

```
# cd /
# tar cf /sys/drivers/console.tar /sys/drivers/console
# patch -p -E < xc/programs/Xserver/hw/sunLynx/patch.Console
# cd /sys/drivers/console
# make install
# cd /sys/lynx.os
# make install
# reboot -a
```

If you opt not to install the patch you must edit `xc/config/cf/lynx.cf` and change the definition of `SparcConsoleDefines`

```
from

        #define SparcConsoleDefines    -DPATCHED_CONSOLE

to

        #define SparcConsoleDefines    /* -DPATCHED_CONSOLE */
```

7.2. Known Bug of the microSPARC server

On the first start of the X server on the microSPARC you will notice that the pointer follows mouse movements with a certain delay (especially if you're moving the mouse real fast). You will also notice that moving windows with certain window managers (eg mwm) is not working correctly. These effects should go away on the next server start.

The server for monochrome cards builds properly if you enable it in `lynx.cf` but it has never been tested (reports are welcome).

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/LynxOS.sgml,v 3.14.2.10 1999/06/02
07:51:49 hohndel Exp $
```

TOC

$XConsortium: LynxOS.sgml /main/10 1996/10/28 05:13:07 kaleb $

Building XFree86

David Dawes

25 June 1999

---

*This document describes how to build XFree86 from the **source** distribution. It covers building from the full source distribution as well as from the cut-down source distribution available for building only the X servers. It is designed to be used in conjunction with the OS-specific README files.*

---

1. Building XFree86 From a Source Distribution

1.1. How to get the XFree86 3.3.4 source

1.2. Configuring the source before building

1.3. Building and installing the distribution

2. Reconfiguring the server (source distribution)

3. Reconfiguring the server (binary distribution)

---

1. Building XFree86 From a Source Distribution

> **NOTE:** Refer to the appropriate OS-specific README file before attempting to build XFree86. These files contain additional information that you may need to successfully build under your OS.

We highly recommend using GCC-2 to build XFree86. GCC-2 is available from prep.ai.mit.edu and other sites archiving GNU source. Note that both gcc-2.8.0 and egcs have been proven to break the code multiple times. Especially egcs seems to fail in several modules when optimizing.

1.1. How to get the XFree86 3.3.4 source

There are a few starting points for getting the XFree86 source. One option is to start directly with the XFree86 3.3.4 source distribution. In this case, the procedure is as follows:

- The XFree86 3.3.4 source is contained in files `X334src-1.tgz`, `X334src-2.tgz` and `X334src-3.tgz`. These can be found at ftp://ftp.xfree86.org/pub/XFree86/3.3.4/source/ and similar locations on XFree86 mirror sites. `X334src-2.tgz` contains the fonts and documentation source. `X334src-3.tgz` contains the hardcopy documentation. `X334src-1.tgz` contains everything else. If you don't need the docs or fonts you can get by with only `X334src-1.tgz`.
- Extract each of these files by running the following from a directory on a filesystem containing enough space (the full source requires around 140MB, and a similar amount is required in addition to this for the compiled binaries):

```
gzip -d < X334src-1.tgz | tar vxf -
gzip -d < X334src-2.tgz | tar vxf -
gzip -d < X334src-3.tgz | tar vxf -
```

Another option is to start with the X11R6.3 source distribution and patch it up to XFree86 3.3.3 and then patch that to XFree86 3.3.4 (see below). In this case you need to do the following:

- Start with the X Consortium's X11R6.3 distribution with public patches 1 and 2 (but not 3) applied. This can be obtained by following the links from the The Open Group's X home page.
- Get the files `R6.3pl2-3.3.3.diff1.gz`, `R6.3pl2-3.3.3.diff2.gz`, `R6.3pl2-3.3.3.diff3.gz`, and `R6.3pl2-3.3.3.diff4.gz` from ftp://ftp.xfree86.org/pub/XFree86/3.3.3/patches/ (or a similar location on mirror sites). To upgrade the source to XFree86 3.3.3, run the following from directory containing the `xc` directory of the X11R6.3 pl2 source tree:


```
gzip -d < R6.3pl2-3.3.3.diff1.gz | patch -p0 -E
gzip -d < R6.3pl2-3.3.3.diff2.gz | patch -p0 -E
gzip -d < R6.3pl2-3.3.3.diff3.gz | patch -p0 -E
gzip -d < R6.3pl2-3.3.3.diff4.gz | patch -p0 -E
```

Be sure to do this with a clean unmodified source tree. If you don't some patches may fail.

A further option is to start with the XFree86 3.3.3 source, and patch it up to XFree86 3.3.4. In this case you need to do the following:

- If using this option, you would already have the XFree86 3.3.3 source. If you have applied any of the public patches to 3.3.3, back them out before starting the upgrade to 3.3.4.
- Get the file `3.3.3-3.3.4.diff.gz` from ftp://ftp.xfree86.org/pub/XFree86/3.3.4/patches/ (or a similar location on mirror sites). To upgrade the source to XFree86 3.3.4, run the following from directory containing the `xc` directory of the XFree86 3.3.3 source tree:

```
gzip -d < 3.3.3-3.3.4.diff.gz | patch -p0 -E
```

Be sure to do this with a clean unmodified source tree. If you don't some patches may fail.

If you only want to build the XFree86 X servers, you can use a cut-down version of the XFree86 source tree called the ``servers only'' distribution. If you choose this option, do the following:

- Get the `X334servonly.tgz` file from ftp://ftp.xfree86.org/pub/XFree86/3.3.4/source/ (or a similar locations on mirror sites.
- Extract this by running the following:

```
gzip -d < X334servonly.tgz | tar vxf -
```

XFree86 supports a small subset of the X Consortium X11R6.1 contrib distribution. If you wish to build this, you will need at least the following files/directories from that distribution:

```
contrib/Imakefile
contrib/programs/Imakefile
contrib/programs/ico
contrib/programs/listres
contrib/programs/showfont
contrib/programs/viewres
contrib/programs/xbiff
contrib/programs/xcalc
contrib/programs/xditview
contrib/programs/xedit
contrib/programs/xev
contrib/programs/xeyes
contrib/programs/xfontsel
```


```
contrib/programs/xgc
contrib/programs/xload
contrib/programs/xman
contrib/programs/xmessage
```

You will also need the XFree86 patch `contrib-3.3.3.diff.gz`. To apply the patch, run the following from the directory containing the `contrib` directory:

```
gzip -d < contrib-3.3.3.diff.gz | patch -p0 -E
```

Alternatively, you can just get the file `X333contrib.tgz` from the XFree86 source directory, and extract it by running:

```
gzip -d < X333contrib.tgz | tar vxf -
```

If you wish to build the xtest distribution, get the source distribution `X33test.tgz` from the XFree86 source directory, and extract it by running:

```
gzip -d < X33test.tgz | tar vxf -
```

Note, xtest is no longer part of the core X11 distribution (since X11R6.3).

1.2. Configuring the source before building

It is recommended that you start the configuration process by going to the `xc/config/cf` directory, and copying the file `xf86site.def` to `host.def`. Then read through the `host.def` file (which is heavily commented), and set any parameters that you want for your configuration. You can usually find out what the default settings are by checking the `.cf` file(s) relevant to your OS.

Unlike previous versions, imake can now automatically detect and set the various **OS*Version** parameters, so you shouldn't need to enter those settings explicitly.

If you are using just the `X334src-1.tgz` part of the source dist, you will need to define **BuildFonts** to **NO**.

If you are using the ``servers only'' distribution, you will need to define **BuildServersOnly** to **YES**.

1.3. Building and installing the distribution

Before building the distribution, read through the OS-specific `README` file in `xc/programs/Xserver/hw/xfree86/doc` that is relevant to you. Once those OS-specific details have been taken care of, go the `xc` directory and run ``make World'' with the **BOOTSTRAPCFLAGS** set as described in the OS-specific README (if necessary). It is advisable to redirect stdout and stderr to `World.Log` so that you can track down problems that might occur during the build.

When the build is finished, you should check `World.Log` to see if there were any problems. If there weren't any then you can install the binaries. When using the full source distribution, the installation should be done from the `xc` directory. When using the ``servers only'' distribution, the install should be done from the `xc/programs/Xserver` directory. To do the install, run ``make install'' and

``make install.man''. Make sure you have enough space in /usr/X11R6 for the install to succeed. If you want to install on a filesystem other than /usr, make a symbolic link to /usr/X11R6 before installing.

To install the binary LinkKit (in /usr/X11R6/lib/Server), run ``make install.linkkit'' from the xc directory.

To build the subset of the contrib release supported by XFree86, make sure that you have first built and installed the core distribution. Then go to the contrib directory and run ``xmkmf -a; make''. When that is completed, run ``make install'' and ``make install.man'' to install it.

To build/run the xtest distribution, refer to the instructions in the file test/xsuite/NOTES.xf86.

Building XFree86 **:** *Building XFree86 From a Source Distribution*
Previous: *Building XFree86*
Next: *Reconfiguring the server (source distribution)*

2. Reconfiguring the server (source distribution)

To build a different set of servers or servers with a different set of drivers installed:

1. Make sure the source for any new drivers is in the correct place (e.g., driver source for the SVGA server should be in a subdirectory of `xc/programs/Xserver/hw/xfree86/vga256/drivers`).
2. Change the settings of the server defines in `host.def` to specify which servers you wish to build. Also, change the driver lists to suit your needs.
3. From `xc/programs/Xserver`, run:

```
make Makefile
make Makefiles
make depend
make
```

3. Reconfiguring the server (binary distribution)

If you have installed the server Binary LinkKit, it is possible to reconfigure the drivers and some of the extensions in the servers. For details of how to do this, please refer to the README.LinkKit file.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/BUILD.sgml,v 3.1.2.10 1999/06/25
08:57:13 hohndel Exp $
```

Readme for the XFree86 3.3.3 LinkKit

TOC

The XFree86 Project, Inc.

22 October 1998

1. Readme for the XFree86 3.3.3 LinkKit

1. Readme for the XFree86 3.3.3 LinkKit

1. For systems which don't use gcc-2, you may need to install libgcc.a if the binary distribution you are using was built with gcc-2.
2. Make sure that you have the XFree86 3.3.3 libraries installed under `/usr/X11R6` if you will be linking Xnest with the LinkKit. The LinkKit is now self-contained for the other servers.
3. Edit the `xf86site.def` file to define which servers you want to build, and the drivers and extensions you want included.
   - Set `HasGcc` and `HasGcc2` to match the compiler you are using if the defaults aren't correct.
   - If the LinkKit was built with gcc-2.x and you are using some other compiler, you must install libgcc.a and set `NeedLibGcc` to `YES`.
   - To build the 256 colour server: set `XF86SVGAServer` to `YES`.
   - To build the 16 colour server: set `XF86VGA16Server` to `YES`.
   - To build the monochrome server: set `XF86MonoServer` to `YES`.
   - To build the S3 server: set `XF86S3Server` to `YES`.
   - To build the Mach8 server: set `XF86Mach8Server` to `YES`.
   - To build the Mach32 server: set `XF86Mach32Server` to `YES`.
   - To build the Mach64 server: set `XF86Mach64Server` to `YES`.
   - To build the P9000 server: set `XF86P9000Server` to `YES`.
   - To build the AGX server: set `XF86AGXServer` to `YES`.
   - To build the ET4000/W32 server: set `XF86W32Server` to `YES`.
   - To build the IBM 8514/A server: set `XF86I8514Server` to `YES`.
   - To build the I128 server: set `XF86I128Server` to `YES`.
   - To build the GLINT server: set `XF86GLINTServer` to `YES`.
   - To build the TGA server: set `XF86TGAServer` to `YES`.
   - To build the GA-98NB/WAP server: set `XF98GANBServer` to `YES`.
   - To build the NEC480 server: set `XF98NEC480Server` to `YES`.
   - To build the NEC-CIRRUS/EPSON NKV/NKV2 server: set `XF98NKVNECServer` to `YES`.
   - To build the WAB-S server: set `XF98WABSServer` to `YES`.
   - To build the WAB-EP server: set `XF98WABEPServer` to `YES`.
   - To build the WSN-A2F server: set `XF98WSNAServer` to `YES`.
   - To build the Trident Cyber9320/9680 server: set `XF98TGUIServer` to `YES`.
   - To build the Matrox Millennium/Mystique Server: set `XF98MGAServer` to `YES`.
   - To build the Cirrus Logic CLGD7555 Server: set `XF98SVGAServer` to `YES`.
   - To build the EGC server: set `XF98EGCServer` to `YES`.
   - To build the NEC S3 server: set `XF98NECS3Server` to `YES`.
   - To build the S3 PW/PCSKB server: set `XF98PWSKBServer` to `YES`.
   - To build the S3 PW/LB server: set `XF98PWLBServer` to `YES`.
   - To build the S3 GA-968 server: set `XF98GA968Server` to `YES`.
   - To build the Xnest server: set `XnestServer` to `YES`.

- If you are building more than one Xserver, uncomment the `ServerToInstall` line and set it to the nam the Xserver you want to be the default server (i.e., the one that the ``X'' sym-link points to).
- Set `XF86SvgaDrivers` to the list of drivers you want to include in the SVGA server.
- Set `XF86Vga16Drivers` to the list of drivers you want to include in the 16 colour server.
- Set `XF86Vga2Drivers` to the list of drivers you want to include in the monochrome vga server.
- Set `XF86MonoDrivers` to the list of non-vga mono drivers you want to include in the mono or VGA16 servers (when building dual-headed servers).
- Note: the ordering of drivers determines the order in which the probing is done. The `generic' driver should be the last one included in the Mono and VGA16 and SVGA servers because its probe always succeeds.
- To use dynamically loadable modules(e.g. PEX, XIE): set `ExtensionsDynamicModules` to `YES`.
- To include the PEX extension: set `BuildPexExt` to `YES`.
- To include the X Image Extension: set `BuildXIE` to `YES`.
- To include the GLX Extension: set `BuildGlxExt` to `YES`.
- To exclude the Double Buffer Extension: set `BuildDBE` to `NO`.
- To exclude the Record Extension: set `BuildRECORD` to `NO`.
- To exclude the Screen Saver extension: set `BuildScreenSaverExt` to `NO`.
- Although it is possible to disable other extensions through this mechanism, doing so is not recommended because savings in server size are not appreciable, or the resulting server might even be crippled in some way.

4. If you are including a driver that it not part of the standard distribution, make a directory in `drivers/vga256` (`drivers/vga2` if it is for the VGA2 part of the Mono server, `drivers/vga16` if it is for the 16 colour server, or `drivers/mono` non-VGA part of the Mono and VGA16 servers) and copy either the source or .o file and a suitable Imakefile into that directory. The name of the directory should be the same as the name of the driver (refer to the documentation in the `VGADriverDoc` directory for more details).
5. To build the Makefile, run

   ```
   ./mkmf
   ```
6. Run `make' to link the server(s) as configured.
7. Run `make install' to install the new server(s).
8. Run `make clean' to remove the files that were created by this procedure.
9. If you edit the `xf86site.def` file and change the selection of servers being built or the drivers included in them, repeat the above procedure. If changing the selection of Xserver extensions being included it is sufficient to run `make Makefile' instead of `./mkmf'.
10. It is possible to see which drivers are included in the Xserver by running it with the `-showconfig' flag. To check which extensions are included, start the Xserver and run `xdpyinfo'.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/LinkKit.sgml,v 3.14.2.5 1998/10/22
04:31:03 hohndel Exp $
```

```
$XConsortium: LinkKit.sgml /main/8 1996/10/27 11:05:58 kaleb $
```

README for XFree86 on NetBSD

Rich Murphey, David Dawes, Marc Wandschneider, Mark Weaver, Matthieu Herrb

Last modified on: 20 August 1999

1. What and Where is XFree86?
2. Bug Reports for This Document
3. New OS-dependant features in XFree86 3.3.5
4. New OS-dependant features in XFree86 3.3.4
5. Installing the Binaries
6. Configuring X for Your Hardware
   - 6.1. About mouse configuration
   - 6.2. Other input devices
   - 6.3. Configuring PEX and XIE extensions
7. Running X
   - 7.1. Starting Xdm, the display manager
8. Kernel Support for X
   - 8.1. Console drivers
   - 8.2. Aperture Driver
   - 8.3. MIT-SHM
9. Rebuilding the XFree86 Distribution
   - 9.1. Console drivers
   - 9.2. pcvt_ioctl.h file:
   - 9.3. console.h and ioctl_pc.h files:

9.4. Support for shared libs under NetBSD 1.0 and later
9.5. Building on other architectures

10. Building New X Clients

11. Thanks

TOC

1. What and Where is XFree86?

XFree86 is a port of X11R6.3 that supports several versions of Intel-based Unix. It is derived from X386 1.2, which was the X server distributed with X11R5. This release consists of many new features and performance improvements as well as many bug fixes. The release is available as source patches against the X Consortium X11R6.3 code, as well as binary distributions for many architectures.

See the Copyright Notice.

The sources for XFree86 are available by anonymous ftp from:

ftp://ftp.XFree86.org/pub/XFree86/current

Binaries for NetBSD 1.4 and later are available from:
ftp://ftp.XFree86.org/pub/XFree86/current/binaries/NetBSD-1.4

A list of mirror sites is provided by ftp://ftp.XFree86.org/pub/XFree86/MIRRORS

Other NetBSD versions:

These binaries are not compatible with earlier NetBSD versions. If you're still running NetBSD earlier than 1.4, you should think about upgrading to a newer version of NetBSD first.

If you don't upgrade, you'll have to build XFree86 from the sources. XFree86 should compile cleanly under earlier versions of NetBSD, although this has not been tested.

XFree86 also builds on NetBSD/sparc. See section Building on other architectures for details.

The client side of XFree86 also builds on NetBSD/alpha and many other architecture supported by NetBSD.

XFree86 also supports NetBSD on PC98 machines.

2. Bug Reports for This Document

Send email to *matthieu@laas.fr* (Matthieu Herrb) or *XFree86@XFree86.org* if you have comments or suggestions about this file and we'll revise it.

3. New OS-dependant features in XFree86 3.3.5

None.

See the Release Notes for non-OS dependent new features in XFree86 3.3.5.

4. New OS-dependant features in XFree86 3.3.4

- The maximum number of open connections in the server has been raised to 128,
- support for the **wsmouse** mouse protocol included in NetBSD 1.4 has been added.

5. Installing the Binaries

Refer to section 5 of the Release Notes for detailed installation instructions.

6. Configuring X for Your Hardware

The `/etc/XF86Config` file tells the X server what kind of monitor, video card and mouse you have. You *must* create it to tell the server what specific hardware you have.

The easiest way to create this file is to run the **XF86Setup** utility as root. Refer to QuickStart.doc for details about its use.

You'll need info on your hardware:

- Your mouse type, baud rate and its /dev entry.
- The video card's chipset (e.g. ET4000, S3, etc).
- Your monitor's sync frequencies.

The recommended way to generate an `XF86Config` file is to use the `XF86Setup` utility. The xf86config text utility is still there for the (few) cases where XF86Setup can't be used. Also, there is a sample file installed as `/usr/X11R6/lib/X11/XF86Config.eg`, which can be used as a starting point.

For details about the `XF86Config` file format, refer to the *XF86Config(5)* manual page.

Once you've set up a XF86Config file, you can fine tune the video modes with the `xvidtune` utility.

6.1. About mouse configuration

If your serial mouse does not work try using `kermit` or `tip` to connect to the mouse serial port and verify that it does indeed generate characters.

The NetBSD pms mouse driver handles PS/2 style mice as Busmouse. Specify the protocol as ``**busmouse**'' in the mouse section of your `XF86Config` file if you're using a PS/2 mouse.

Only standard PS/2 mice are supported by this driver. Newest PS/2 mice that send more than three bytes at a time (especially Intellimouse, or MouseMan+ with a ``3D'' roller) are not supported yet.

XFree86 3.3.4 and later also have support for the mouse driver included in the new **wscons** console driver introduced by NetBSD 1.4. Specify ```wsmouse`'' as the protocol and ```/dev/wsmouse0`'' as the device in `/etc/XF86Config` if you're using NetBSD 1.4 with wscons.

See README.mouse for general instruction on mouse configuration in XFree86.

6.2. Other input devices

XFree86 supports the dynamic loading of drivers for external input devices using the `XInput` extension. Currently supported devices are:

- Joystick (`xf86Jstk.so`)
- Wacom tablets (Wacom IV protocol only, `xf86Wacom.so`)
- SummaSketch tablets (`xf86Summa.so`)
- Elographics touchscreen (`xf86Elo.so`)

To use a specific device, add the line

```
load "module"
```

in the **Module** section of `XF86Config`, where *module* is the name of the `.so` file corresponding to your device. You also need to set up a **XInput** section in `XF86Config`. Refer to the *XF86Config(5)* man page for detailed configuration instructions.

You can then change the device used to drive the X pointer with the *xsetpointer(1)* command.

For joystick support, you'll need to install the joystick device driver in the kernel. It is included in NetBSD 1.2. See *joy(4)* for details.

6.3. Configuring PEX and XIE extensions

The PEX and XIE extensions are supported as external modules. If you want to have access to these extensions, add the following lines to the **Module** section of `XF86Config`:

```
load    "pex5.so"
load    "xie.so"
```

7. Running X

8mb of memory is a recommended minimum for running X. The server, window manager and an xterm take about 4 Mb of memory themselves. On a 4Mb system that would leave nothing left over for other applications like gcc that expect a few meg free. X will work with 4Mb of memory, but in practice compilation while running X can take 5 or 10 times as long due to constant paging.

The easiest way for new users to start X windows is to type:

```
startx >& startx.log
```

Error messages are lost unless you redirect them because the server takes over the screen.

To get out of X windows, type: ``exit'' in the console xterm. You can customize your X by creating `.xinitrc`, `.xserverrc`, and `.twmrc` files in your home directory as described in the xinit and startx man pages.

7.1. Starting Xdm, the display manager

To start the display manager, log in as root on the console and type: ``xdm -nodaemon''.

You can start xdm automatically on bootup by disabling the console getty and adding the following code to `/etc/rc.local`:

```
if [ -x /usr/X11R6/bin/xdm ]; then
  echo -n ' xdm'; /usr/X11R6/bin/xdm
fi
```

To disable the console getty, change ``**on**'' to ``**off**'' in the console entry in `/etc/ttys`:

```
ttyv0  "/usr/libexec/getty Pc" pc      off secure
```

Under NetBSD 1.4 with the wscons console driver, you must enable a virtual console for the X server first. To do this follow these steps:

- Make sure the device file exists. If not, ``cd /dev ; ./MAKEDEV wscons''.
- Next, make sure your kernel wants to do wscons. (see below).
- Next, make sure ``wscons=YES'' in `/etc/rc.conf`.
- Next, make sure `/etc/wscons.conf` exists. The relevant bits:

```
#screen 0       -       vt100
```

TOC

```
screen  1       -       vt100
screen  2       -       vt100
screen  3       -       vt100
screen  4       -       -
screen  5       -       vt100
```

(Thanks to Mason Loring Bliss <mason@acheron.middleboro.ma.us> for this explanation)

Note that the binary distributions of XFree86 for NetBSD don't include support for the XDM-AUTHORIZATION-1 protocol.

README for XFree86 on NetBSD **:** *Running X*
Previous: *Configuring X for Your Hardware*
Next: *Kernel Support for X*

8. Kernel Support for X

To make sure X support is enabled under NetBSD, the following line must be in your config file in `/sys/arch/i386/conf`:

```
options XSERVER, UCONSOLE
```

8.1. Console drivers

The server supports the standard NetBSD/i386 console drivers: pccons, pcvt and wscons (in pcvt compatibility mode). They are detected at runtime and no configuration of the server itself is required.

The pccons driver is the most widely tested and is the console driver contained in the NetBSD binary distribution's kernels.

The pcvt console driver is bundled with NetBSD. The pcvt X mode is compatible with the pccons driver X mode. It offers several virtual consoles and international keyboard support. In order to use this driver, change the line:

```
device pc0 at isa? port "IO_KBD" irq 1
```

to

```
device vt0 at isa? port "IO_KBD" irq 1
```

in your kernel config file, and rebuild and install your kernel.

XFree86 will also run with the wscons console driver in NetBSD 1.4. For now, it uses the pcvt compatibility mode, so be sure to have the lines:

```
options         WSDISPLAY_COMPAT_PCVT           # emulate some ioctls
options         WSDISPLAY_COMPAT_SYSCONS        # emulate some ioctls
options         WSDISPLAY_COMPAT_USL            # VT handling
options         WSDISPLAY_COMPAT_RAWKBD         # can get raw scancodes
```

in your kernel configuration file if you're using wscons. Refer to the *wscons(4)* and *wsmouse(4)* manual pages for informations on how to configure wscons into the kernel.

8.2. Aperture Driver

By default NetBSD 0.9C and higher include the BSD 4.4 kernel security feature that disable access to the `/dev/mem` device when in multi-users mode. But XFree86 servers can take advantage (or require) linear access to the display memory.

Moset recent accelerated servers require linear memory access, some other can take advantage of it, but do not require it.

There are two ways to allow XFree86 to access linear memory:

The first way is to disable the kernel security feature by adding `option INSECURE' in the kernel configuration file and build a new kernel.

The second way is to install the aperture driver: You can get the aperture driver sources from ftp://ftp.netbsd.org/pub/NetBSD/arch/i386/apNetBSD.shar.

How to activate it is highly dependent from your exact operating system version:

- NetBSD 1.0, 1.1, 1.2, 1.2.1:

  Add the following lines to the end of `/etc/rc.local`:

```
KERNDIR=/usr/X11R6/lib/X11/kernel
if [ -f ${KERNDIR}/ap.o ]; then
  modload -o ${KERNDIR}/ap -e ap -p ${KERNDIR}/apinstall ${KERNDIR}/ap.o
fi
```

- NetBSD 1.2D and later

  Add the following line to `/etc/lkm.conf`:

```
/usr/X11R6/lib/X11/kernel/ap.o  -  ap /usr/X11R6/lib/X11/kernel/apinstall -
```

- NetBSD 1.2G, 1.3 and later

  The `lkm.conf` format changed in 1.2G. Add the following line to `/etc/lkm.conf`:

```
/usr/X11R6/lib/X11/kernel/ap.o  -  ap /usr/X11R6/lib/X11/kernel/apinstall -
-AFTERMOUNT
```

Reboot your system. XFree86 will auto-detect the aperture driver if available.

Warning: if you boot another kernel than `/netbsd` or `/bsd`, loadable kernel modules can crash your system. Always boot in single user mode when you want to run another kernel.

> **Caveat:** the aperture driver only allows one access at a time (so that the system is in the same security state once X is launched). This means that if you run multiple servers on multiples VT, only the first one will have linear memory access. Use 'option INSECURE' if you need more that one X server at a time.

8.3. MIT-SHM

NetBSD 1.0 and later supports System V shared memory. If XFree86 detects this support in your kernel, it will support the MIT-SHM extension.

To add support for system V shared memory to your kernel add the lines:

```
# System V-like IPC
options         SYSVMSG
options         SYSVSEM
options         SYSVSHM
```

to your kernel config file.

9. Rebuilding the XFree86 Distribution

The server link kit allow you to rebuild just the X server with a minimum amount of disk space. Just unpack it, make the appropriate changes to the `xf86site.def`, type ``./mkmf'' and ``make'' to link the server. See `/usr/X11R6/lib/Server/README` for more info.

See INSTALL for instructions on unbundling and building the source distribution.

You should configure the distribution by editing `xc/config/cf/host.def` before compiling. To compile the sources, invoke ``make World'' in the xc directory.

9.1. Console drivers

XFree86 has a configuration option to select the console drivers to use in `xf86site.def`:

- if you're using pccons put:

```
#define XFree86ConsoleDefines -DPCCONS_SUPPORT
```

- if you're using pcvt put:

```
#define XFree86ConsoleDefines -DPCVT_SUPPORT
```

- if you're using syscons put:

```
#define XFree86ConsoleDefines -DSYSCONS_SUPPORT
```

- if you're running codrv put:

```
#define XFree86ConsoleDefines -DCODRV_SUPPORT
```

If you don't define **XFree86ConsoleDefines** in `xf86site.def` the pccons and pcvt drivers will be supported by default.

Syscons and codrv are not bundled with NetBSD. They are available by anonymous FTP from a number of sites. They are not supported by the XFree86 binary distribution anymore.

9.2. pcvt_ioctl.h file:

XFree86's default configuration includes support for the PCVT console driver. Unfortunately, NetBSD versions before 19980413 don't install the `pcvt_ioctl.h` file in `/usr/include/machine`. If you want to build XFree86 with PCVT support, execute the following command as root before starting `make World`:

```
cp /usr/src/sys/arch/i386/isa/pcvt/pcvt_ioctl.h /usr/include/machine
```

If you don't have kernel sources, you can grab this file from ftp.netbsd.org or one of its mirrors. If you r running PCVT, you can remove -DPCVT_SUPPORT from **XFree86ConsoleDefines** in `xf86site.def` too.

9.3. console.h and ioctl_pc.h files:

If you want to build a server supporting codrv and you don't already have the corresponding header file `ioctl_pc.h` installed in `/usr/include/machine`, then install the copy that is supplied in `xc/programs/Xserver/hw/xfree86/etc`. If you run NetBSD-current you probably want to install it in `/usr/src/sys/arch/i386/include` too, so that it get reinstalled each time you run `make includes`.

If you have installed the codrv console driver, this file should be taken from your installed version of the driver.

The `console.h` file for syscons isn't distributed with XFree86 anymore. You should get it from the syscons distribution.

9.4. Support for shared libs under NetBSD 1.0 and later

By default XFree86 builds for NetBSD with shared libraries support. If you're building on 0.9 or don't want shared libraries add the following line to `host.def`:

```
#define BuildBsdSharedLibs NO
```

9.5. Building on other architectures

XFree86 also compiles on NetBSD/sparc. The Sun server patches from Dennis Ferguson and Matthew Green have been integrated in `xc/programs/Xserver/hw/sun`. Small modifications to `xf86site.def` are needed:

- Set all variables defining the servers to build to **NO**. (The variables controlling the Sun servers to build **Xsun24Server**, **XsunServer** and **XsunMonoServer** are defined at the end of `NetBSD.cf`.)
- Set **ServerToInstall** to the sun server of your choice. (Xsun or XsunMono).
- Look at other applicable options in the INSTALL document.

Problems with this port should be reported to the *port-sparc@NetBSD.Org* mailing list or directly to me *matthieu@laas.fr* rather than to the xfree86 mailing list.

Note that the NetBSD project has now its own source tree, based on the XFree86 source tree, with some local modifications. You may want to start with this tree to rebuild from sources. The NetBSD xsrc source tree is available at: ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-current/xsrc/

10. Building New X Clients

The easiest way to build a new client (X application) is to use `xmkmf` if an `Imakefile` is included in the sources. Type ``xmkmf -a'' to create the Makefiles, check the configuration if necessary and type ``make''. Whenever you install additional man pages you should update `whatis.db` by running ``makewhatis /usr/X11R6/man''.

To avoid the ``Virtual memory exhausted'' message from cc while compiling, increase the data and stack size limits (in csh type ``limit datasize 32M'' and ``limit stacksize 16M'').

Note: Starting with XFree86 2.1 and NetBSD 0.9A, the symbol **__386BSD__** no longer gets defined either by the compiler or via the X config files for *BSD systems. When porting clients to *BSD systems, make use of the symbol **BSD** for code which is truly BSD-specific. The value of the symbol can be used to distinguish different BSD releases. For example, code specific to the Net-2 and later releases can use:

```
#if (BSD >= 199103)
```

To ensure that this symbol is correctly defined, include `<sys/param.h>` in the source that requires it. Note that the symbol **CSRG_BASED** is defined for *BSD systems in XFree86 3.1.1 and later. This should be used to protect the inclusion of `<sys/param.h>`.

For code that really is specific to a particular i386 BSD port, use **__FreeBSD__** for FreeBSD, **__NetBSD__** for NetBSD, **__OpenBSD__** for OpenBSD, **__386BSD__** for 386BSD, and **__bsdi__** for BSD/386.

Another note: If you get the message:

```
ld.so: undefined symbol _XtCvtStringToFont
```

at run-time, you've stumbled on a semantic weakness of the NetBSD dynamic linker. Applications that use libXmu also need libXt. If the client uses a standard `Imakefile`, this dependency will probably by included in the Makefile automagically -- you'll not see the problem. Otherwise, just add ``-lXt'' to your library list in the Imakefile or Makefile and relink.

11. Thanks

Many thanks to:

- **Pace Willison** for providing the initial port to 386BSD.
- **Amancio Hasty** for fixing cursor restoration, mouse bugs and many others.
- **Christoph Robitschko** for fixing `com.c` and thus select().
- **Nate Williams** for the patchkit support for X.
- **Rod Grimes** and **Jack Velte** of Walnut Creek Cdrom for use of their machines in preparing the FreeBSD binary release.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/NetBSD.sgml,v 3.45.2.13 1999/08/23
08:49:48 hohndel Exp $
```

```
$XConsortium: NetBSD.sgml /main/26 1996/10/28 05:43:20 kaleb $
```

README for XFree86 on OpenBSD

Matthieu Herrb

Last modified on: 20 August 1999

1. What and Where is XFree86?
2. Bug Reports for This Document
3. New OS-dependent features in this release
4. New OS-dependent features in XFree86 3.3.4
5. Installing the Binaries
6. Configuring X for Your Hardware
   - 6.1. About mouse configuration
   - 6.2. Other input devices
   - 6.3. Configuring PEX and XIE extensions
7. Running X
   - 7.1. Starting xdm, the display manager
   - 7.2. Running X without the display manager
8. Kernel Support for X
   - 8.1. Console drivers
   - 8.2. Aperture Driver
   - 8.3. MIT-SHM
9. Rebuilding the XFree86 Distribution
   - 9.1. Console drivers
   - 9.2. Building on other architectures

10. Building New X Clients

TOC

1. What and Where is XFree86?

XFree86 is a port of X11R6.3 that supports several versions of Intel-based Unix. It is derived from X386 1.2, which was the X server distributed with X11R5. This release consists of many new features and performance improvements as well as many bug fixes. The release is available as source patches against the X Consortium X11R6.3 code, as well as binary distributions for many architectures.

See the Copyright Notice.

The sources for XFree86 are available by anonymous ftp from:

ftp://ftp.XFree86.org/pub/XFree86/current

Binaries for OpenBSD 2.5 and later are available from:

ftp://ftp.XFree86.org/pub/XFree86/current/binaries/OpenBSD

A list of mirror sites is provided by ftp://ftp.XFree86.org/pub/XFree86/MIRRORS

XFree86 also builds on other OpenBSD architectures. See section Building on other architectures for details.

2. Bug Reports for This Document

Send email to *matthieu@laas.fr* (Matthieu Herrb) or *XFree86@XFree86.org* if you have comments or suggestions about this file and we'll revise it.

3. New OS-dependent features in this release

None. See the Release Notes for non-OS dependent new features in XFree86 3.3.5.

4. New OS-dependent features in XFree86 3.3.4

- The maximum number of open connections in the server has been raised to 128,
- the `resize` utility was fixed.

5. Installing the Binaries

Refer to section 5 of the Release Notes for detailed installation instructions.

6. Configuring X for Your Hardware

The `/etc/XF86Config` file tells the X server what kind of monitor, video card and mouse you have. You *must* create it to tell the server what specific hardware you have.

The easiest way to create this file is to run the **XF86Setup** utility as root. Refer to QuickStart.doc for details about its use.

You'll need info on your hardware:

- Your mouse type, baud rate and its /dev entry.
- The video card's chipset (e.g. ET4000, S3, etc).
- Your monitor's sync frequencies.

The recommended way to generate an `XF86Config` file is to use the `XF86Setup` utility. The xf86config text utility is still there for the (few) cases where XF86Setup can't be used. Also, there is a sample file installed as `/usr/X11R6/lib/X11/XF86Config.eg`, which can be used as a starting point.

For details about the `XF86Config` file format, refer to the *XF86Config(5)* manual page.

Once you've set up a XF86Config file, you can fine tune the video modes with the `xvidtune` utility.

6.1. About mouse configuration

If your serial mouse does not work try using `kermit` or `tip` to connect to the mouse serial port and verify that it does indeed generate characters.

The OpenBSD pms driver provides both ``raw'' and ``cooked'' (translated) modes. ``raw'' mode does not do protocol translation, so XFree86 would use the **PS/2** protocol for talking to the device in that mode. ``Cooked'' mode is the old BusMouse translation. The driver runs in ``raw'' mode when using the `/dev/psm0` device name.

On OpenBSD 2.2, only standard PS/2 mice are supported by this driver.

On OpenBSD 2.3 and later include there is support for recent PS/2 mice that send more than three bytes at a time (especially intellimouse, or mouseman+ with a "3D" roller).

See README.mouse for general instruction on mouse configuration in XFree86.

6.2. Other input devices

XFree86 supports the dynamic loading of drivers for external input devices using the `XInput` extension. Currently supported devices are:

- Joystick (`xf86Jstk.so`)
- Wacom tablets (Wacom IV protocol only, `xf86Wacom.so`)
- SummaSketch tablets (`xf86Summa.so`)
- Elographics touchscreen (`xf86Elo.so`)

To use a specific device, add the line

```
load "module"
```

in the **Module** section of `XF86Config`, where *module* is the name of the `.so` file corresponding to your device. You also need to set up a **XInput** section in `XF86Config`. Refer to the *XF86Config(5)* man page for detailed configuration instructions.

You can then change the device used to drive the X pointer with the *xsetpointer(1)* command.

For joystick support, you'll need to enable the joystick device driver in the kernel. See *joy(4)* for details.

6.3. Configuring PEX and XIE extensions

The PEX and XIE extensions are supported as external modules. If you want to have access to these extensions, add the following lines to the **Module** section of `XF86Config`:

```
load    "pex5.so"
load    "xie.so"
```

README for XFree86 on OpenBSD : *Configuring X for Your Hardware*
Previous: *Installing the Binaries*
Next: *Running X*

7. Running X

8mb of memory is a recommended minimum for running X. The server, window manager and an xterm take about 4 Mb of memory themselves. On a 4Mb system that would leave nothing left over for other applications like gcc that expect a few meg free. X will work with 4Mb of memory, but in practice compilation while running X can take 5 or 10 times as long due to constant paging.

7.1. Starting xdm, the display manager

To start the display manager, log in as root on the console and type: ``xdm -nodaemon''.

You can start xdm automatically on bootup un-commenting the following code in `/etc/rc.local`:

```
if [ -x /usr/X11R6/bin/xdm ]; then
  echo -n ' xdm'; /usr/X11R6/bin/xdm
fi
```

On the default OpenBSD 2.2 installation, you will need to create the virtual console device for the X server:

```
cd /dev
 ./MAKEDEV ttyC5
```

Note that the binary distributions of XFree86 for OpenBSD don't include support for the XDM-AUTHORIZATION-1 protocol.

7.2. Running X without the display manager

The easiest way for new users to start X windows is to type: ``startx >& startx.log''. Error messages are lost unless you redirect them because the server takes over the screen.

To get out of X windows, type: ``exit'' in the console xterm. You can customize your X by creating `.xinitrc`, `.xserverrc`, and `.twmrc` files in your home directory as described in the xinit and startx man pages.

8. Kernel Support for X

To make sure X support is enabled under OpenBSD, the following line must be in your config file in `/sys/arch/i386/conf`:

```
options XSERVER
```

8.1. Console drivers

The server supports the two standard OpenBSD/i386 console drivers: pccons and pcvt. They are detected at runtime and no configuration of the server itself is required.

The pcvt console driver is the default in OpenBSD. It offers several virtual consoles and international keyboard support.

8.2. Aperture Driver

By default OpenBSD includes the BSD 4.4 kernel security feature that disable access to the `/dev/mem` device when in multi-users mode. But XFree86 servers can take advantage (or require) linear access to the display memory.

The P9000, Mach64 and AGX servers require linear memory access, other accelerated servers can take advantage of it, but do not require it. Some drivers in the SVGA server require linear memory access too, notably the Matrox driver.

The preferred way to allow XFree86 to access linear memory is to use the aperture driver

Ths step is highly dependent from your exact operating system version:

- OpenBSD 2.0 Use the aperture driver from /usr/lkm: add the following lines to the end of `/etc/rc.local`:

```
KERNDIR=/usr/lkm
if [ -f ${KERNDIR}/ap.o ]; then
  modload -o ${KERNDIR}/ap -e ap -p \
  ${KERNDIR}/apinstall ${KERNDIR}/ap.o
fi
```

- OpenBSD 2.1, 2.2 Uncomment the lines loading the aperture driver from `/etc/rc.securelevel`
- OpenBSD 2.3 The aperture driver is part of the kernel. Add 'option APERTURE' to your kernel configuration file, build and install the new kernel and run `./MAKEDEV std` in `/dev`. Edit `/etc/sysctl.conf` to set the variable **machdep.allowaperture** to 1.

- OpenBSD 2.4 and later OpenBSD now requires the aperture driver to be enabled for all X serv because the aperture driver also controls access to the I/O ports of the video boards.

After doing that, reboot your system. XFree86 will auto-detect the aperture driver if available.

Warning: if you boot another kernel than `/bsd`, loadable kernel modules can crash your system. Always boot in single user mode when you want to run another kernel.

> **Caveat:** the aperture driver only allows one access at a time (so that the system is in the same security state once X is launched). This means that if you run multiple servers on multiples VT, only the first one will have linear memory access. Use 'option INSECURE' if you need more that one X server at a time.

Another (less recommended) way to enable linear memory access is to disable the kernel security feature by adding ``option INSECURE'' in your kernel configuration file and build a new kernel. In OpenBSD 2.2 and later, you will also need to comment out the line initializing `securelevel` to 1 in `/etc/rc.securelevel`.

8.3. MIT-SHM

OpenBSD supports System V shared memory. If XFree86 detects this support in your kernel, it will support the MIT-SHM extension.

To add support for system V shared memory to your kernel add the lines:

```
# System V-like IPC
options         SYSVMSG
options         SYSVSEM
options         SYSVSHM
```

to your kernel config file.

README for XFree86 on OpenBSD : *Kernel Support for X*
Previous: *Running X*
Next: *Rebuilding the XFree86 Distribution*

9. Rebuilding the XFree86 Distribution

The server link kit allow you to rebuild just the X server with a minimum amount of disk space. Just unpack it, make the appropriate changes to the `xf86site.def`, type ``./mkmf'' and ``make'' to link the server. See `/usr/X11R6/lib/Server/README` for more info.

See INSTALL for instructions on unbundling and building the source distribution.

You should configure the distribution by editing `xc/config/cf/host.def` before compiling. To compile the sources, invoke ``make World'' in the xc directory.

9.1. Console drivers

XFree86 has a configuration option to select the console drivers to use in `xf86site.def`:

- if you're using pccons only put:

```
#define XFree86ConsoleDefines -DPCCONS_SUPPORT
```

- if you're using pcvt only put:

```
#define XFree86ConsoleDefines -DPCVT_SUPPORT
```

If you don't define **XFree86ConsoleDefines** in `xf86site.def` the pccons and pcvt drivers will be supported.

9.2. Building on other architectures

XFree86 also compiles on other OpenBSD architectures.

The XFree86 servers can also been built on OpenBSD/mips. The S3 server has been tested on an Acer Mips system with a S3/928 board. Contact Per Fogelstrom (pefo@OpenBSD.org) for details.

The Xsun server patches from Dennis Ferguson and Matthew Green for NetBSD have been integrated in `xc/programs/Xserver/hw/sun`. The Xsun server can be built on the sparc and the sun3.

The client side of XFree86 also builds on the alpha, pmax, amiga, mac68k and mvme68k architectures.

Problems with this port should be reported directly to the OpenBSD mailing lists rather than to the xfree86 mailing list.

Note that OpenBSD project has now its own source tree, based on the XFree86 source tree, with some local modifications. You may want to start with this tree to rebuild from sources. The OpenBSD X11 source tree is available by anoncvs from all OpenBSD anoncvs servers. See

http://www.openbsd.org/anoncvs.html for details on anoncvs.

TOC

10. Building New X Clients

The easiest way to build a new client (X application) is to use `xmkmf` if an `Imakefile` is included in the sources. Type ``xmkmf -a'' to create the Makefiles, check the configuration if necessary and type ``make''. Whenever you install additional man pages you should update `whatis.db` by running ``makewhatis /usr/X11R6/man''.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/OpenBSD.sgml,v 1.1.2.11 1999/08/23
08:49:49 hohndel Exp $
```

```
$XConsortium$
```

README for XFree86 on OS/2

Holger Veit

Last modified on: August 1st, 1999

1. Introductory Note about the release 3.3.5
2. What and Where is XFree86?
3. Bug Reports for This Document
4. Hardware and Software Requirements
    4.1. Supported, Required, and Recommended Hardware
    4.2. Required Software
5. Installing the System
6. Troubleshooting
7. Checking Compatibility of Video Hardware
8. Installing the packages
9. Adding Variables to CONFIG.SYS
10. Remarks on the Network Configuration
11. Configuring X for Your Hardware
12. Running X
13. Rebuilding the XFree86 Distribution
14. Building New X Clients

15. Acknowledgements

TOC

1. Introductory Note about the release 3.3.5

Before looking into this file, please check for any LATEST.OS2 files that may come with the binary distribution. Please also check out the following XFree86/OS2 WWW pages:

- http://set.gmd.de/~veit/os2/xf86os2.html
- http://set.gmd.de/~veit/os2/xf86bugs.html
- http://set.gmd.de/~veit/os2/x11os2faq.html

before you claim to have found any problems.

This version of the code is called XFree86/OS2 3.3.5. This is a bugfix release for 3.3.3.1 (3.3.4 was never released for XFree86/OS2) which also adds hardware support for some newer cards, including AGP boards. XFree86-3.3.5 contains all security fixes that were released for earlier versions. See the RELNOTES document for details.

XFree86/OS2-3.3.5 is a full, unrestricted version which does not expire, and for which the complete source code is available. In contrast to beta versions, we consider this code as sufficiently stable for use by an end user. Since there have been numerous bugfixes, we recommend this version, even if you had XFree86/OS2 3.3.x before and it worked satisfyingly with your hardware. By the time 3.3.5 is released, the older version 3.3 will be withdrawn, and archives will be updated to this version. There may still be references to 3.3 or 3.3.x still in documents; these apply to 3.3.5 as well, unless otherwise noted.

Previous versions have been tested in a large number of configurations and have been found to be working, with some bugs left, rather flawlessly.

This release is almost complete (with a few exceptions) regarding the X11R6.3 ``core'' distribution. A subset of the ``contrib'' distribution is available from the ported software page http://set.gmd.de/~veit/os2/xf86ported.html

In the past beta testing, it has been found that the software itself is rather stable and does not damage hardware - provided the user does not try to push the builtin limits and change certain configuration parameters which could operate the video hardware out of specs.

However,

- even with a code we consider stable there is no explicit or implicit warranty that certain code works correctly or works at all
- although no damage reports are known, it does not mean that it is impossible to damage hardware with this code; some deeply hidden bugs may still be present in the software.

It is recommended that you backup essential data of your system before installing this software, but this should be your general precautions before ANY installation. No reports exist that a crashing X server itself actively destroys or modifies data, but it is possible in rare cases that the system is left in an

unusable state (video display mode garbled or system unresponsive, not reacting to mouse or keyboa
actions). If you then hard reset or switch off the system, file caches of the operating system might no
written correctly back to disk, thus causing data loss.

README for XFree86 on OS/2 **:** *Introductory Note about the release 3.3.5*
Previous: *README for XFree86 on OS/2*
Next: *What and Where is XFree86?*

2. What and Where is XFree86?

XFree86 is a port of X11R6.3 that supports several versions of Intel-based Unix. It is derived from X386 1.2, which was the X server distributed with X11R5. This release consists of many new features and performance improvements as well as many bug fixes. The release is available as source patches against the X Consortium X11R6 code, as well as binary distributions for many architectures.

XFree86/OS2 is the name of the implementation of XFree86 on OS/2 based systems.

See the Copyright Notice.

Binaries for OS/2 Warp and Merlin are available from: ftp.XFree86.org:/pub/XFree86/3.3.5/OS2

The WWW page http://borneo.gmd.de/~veit/os2/xf86os2.html will usually show more references to FTP or WWW sites to retrieve sources or binaries.

Other versions:

XFree86/OS2 will run on all dialects of Warp 3, including Warp "red spine box", Warp "blue spine box", Warp Connect, Warp Server, and Warp 4.

For Warp 3 installing fixpack level 17 or later is strongly recommended. There have been a few reports that the installation of FP26 causes XFree86 no longer to work, but I am not sure about a real reason. Current fixpacks for Warp 3, like FP36, seem to work well also.

Warp 4 may be used with or without the recent public fixpack.

Please check in all cases a LATEST.OS2 file.

OS/2 2.11 is not supported any longer with this release, due to lack of a working test environment. Consequently, OS/2 SMP 2.11 is not supported either. Warp Server SMP is supported, but SMP does not give significant advantage, other than the general speedup because of multiple processors working. OS/2 versions 1.X are definitely not supported and will never be.

It is possible to build XFree86/OS2 from the sources. Read about this in the document OS2.NOTES.

3. Bug Reports for This Document

Send email to *Holger.Veit@gmd.de* (Holger Veit) or *XFree86@XFree86.org* if you have comments or suggestions about this file and we'll revise it.

4. Hardware and Software Requirements

4.1. Supported, Required, and Recommended Hardware

- At least a 486DX33 with 16MB RAM is required. A Pentium or Pentium Pro and more main memory is recommended. A 386 or a system with 8MB or less memory is an insufficient configuration.
- There are no specific requirements concerning network cards, disk types, or CD ROM equipment; of course the more powerful, the better.
- Depending on the packages installed, a disk space of 20-55MB on a HPFS formatted partition (or a NFS or ext2fs partition natively allowing long filenames) is required. XFree86/OS2 will not run on FAT partitions.
- You need a video card that is supported by XFree86. Refer to the general README document for a list of supported cards. Note that the sets of video cards supported by XFree86 on one hand and OS/2 on the other hand overlap, but do not match exactly, i.e. the fact that your card is supported by OS/2 does not mean it works with XFree86 as well, and vice versa. XFree86 does not use the video services of the OS/2 operating system.

4.2. Required Software

- Any version of Warp 3 with at least fixpack 17, or Warp 4 is required
- XFree86/OS2-3.3.5 may use a local named-pipe connection or a TCP/IP based network connection.
    1. Warp comes with the Internet Access Kit (IAK), which is sufficient. Warp Connect and Warp Server come with a full version of TCP/IP (3.0). Use of this software is preferred over IAK then.
    2. Warp 4 comes with TCP/IP 4.0 which should also work.
    3. There are reports that with EMX 0.9 fix 4, you can also use the new 32 bit IBM TCP/IP 4.1 product.
    4. The old IBM TCP/IP 2.0, that comes with the IBM PMX product may be used with Warp as well, although it is no longer supported by IBM. Please ensure that you have the latest CSDs installed.

  Other versions of TCP/IP, such as FTP's, DEC's, or Hummingbird's TCP/IP versions, as well as IBM TCP/IP 1.X are not supported. Nor does any networking support from DOS (packet drivers, winsock), Netware, or NetBIOS work, and I won't to provide support for that in the future.
- If you want to write or port applications for XFree86, you are encouraged to do so. You will need a complete installation of EMX/gcc 0.9C fix4 or later for doing so. Neither the second (obsolete)

implementation of gcc, nor any commercial package, including Cset/2, VAC++, Borland C++/OS2, Watcom C++, Metaware C, and others, is suitable for porting, because various parts of the X DLLs rely on certain features only present with EMX.

TOC

README for XFree86 on OS/2 **:** *Hardware and Software Requirements*
Previous: *Bug Reports for This Document*
Next: *Installing the System*

5. Installing the System

The binary distribution is composed of a number of zip archives which are the executables, servers, fonts, libraries, include files, man pages, and config files. The full distribution requires about 40-55MB of disk space.

All archives of this alpha version are packed with the `info-zip` utility, which is available under the name `UNZ512X2.EXE` (or a later version) from many OS/2 archives. Please obtain a native OS/2 version of this unpacker. DOS `PKUNZIP` does not work, because it cannot unpack long file names and extended attributes.

At this moment, the distribution covers only the ``core'' distribution which somewhat reduces the usability. Refer to WWW sites and archives listed in the XFree86/OS2 FAQ and elsewhere to obtain pre-built X clients which were ported to XFree86.

The contents of the packages are:

REQUIRED:

Xbase

A special device driver and the SuperProbe program

Xdoc

READMEs and XFree86 specific man pages.

Xbin

all of the executable X client applications and shared libs

Xfnts

the misc and 75dpi fonts

emxrt

Runtime libraries of EMX

Choose at least one of the following to match your hardware:

X8514

the X server for IBM 8514/A and compatible boards

XAGX

the X server for AGX boards

XGlnt


TOC

the X server for Permedia / GLINT boards

XI128

the X server for #9 Imagination 128 boards

XMa32

the X server for ATI Mach32 graphics boards

XMa64

the X server for ATI Mach64 graphics boards

XMa8

the X server for ATI Mach8 graphics boards

XMono

the Monochrome X Server

XP9K

the X server for P9000 based boards

XS3

the X server for S3 based boards (excluding S3 ViRGE)

XS3V

the X server for S3 ViRGE based boards

XSVGA

the 8-bit pseudo-color X server for Super VGA cards

XVG16

the 4-bit pseudo-color X server for VGA & SVGA cards.

XW32

the X server for et4000w32 based boards

OPTIONAL:

Xman

pre-formatted man pages for the X11 interface and clients

Xf100

100dpi fonts

Xfscl

Speedo and Type1 fonts

Xfnon

Japanese, Chinese and other fonts

Xfcyr

Cyrillic fonts

Xfsrv

the font server with man pages.

Xprog

the X11 header files and programmer's utilities for compiling other X applications

Xpex

PEX fonts and libraries required for PEX applications

In order to save space on your disk and reduce net bandwidth, choose the software to obtain carefully. Each X server is an archive of about 1.2MB and occupies 3.0MB on the disk. You won't normally need more than the single Xserver tailored to your video card.

If it is your first time install, get the `Xbase` archive before any of the other packages. This package contains a driver and a test program, which analyzes your video hardware. If this program fails or reports an incompatible hardware, it makes no sense to obtain the other packages in the hope that they would magically work.

README for XFree86 on OS/2 **:** *Installing the System*
Previous: *Hardware and Software Requirements*
Next: *Troubleshooting*

6. Troubleshooting

Surprised to see this section directly in the beginning? We have put it here because chances are best here not to overlook it. This does not mean that you will necessarily encounter trouble when installing XFree86, but be warned: the following sections are **IMPORTANT** and neglecting one or more things out of impatience or sloppiness will leave you with a non-working X11 system and us with unnecessary problems.

Still, due to the incredibly large number of hardware configurations, there may be some special situations and configurations where the below description is not successful. If this happens, read - I repeat **READ** - the list of ``frequently asked questions'' (FAQ) which has meanwhile evolved to a troubleshooting guide. The latest version is always at http://set.gmd.de/~veit/os2/x11os2faq.html .

Maybe - but we found you must be very creative - you find a bug. Consult the page http://set.gmd.de/~veit/os2/xf86bugs.html whether it is already known. If not, you have a case and should report it to XFree86 (*xfree86@xfree86.org*). Please refer to the FAQ about the information to be provided for a complete problem report.

The recommended newsgroup for setup questions is *comp.os.os2.setup.misc*. I read this group, so it won't speed up the process or enforce anything if you post to other groups, or forward the report to my mail address as well or to xfree86@xfree86.org.

So, not to discourage you completely, the setup section begins:

7. Checking Compatibility of Video Hardware

In the following, we assume that you want to install XFree86/OS2 on a disk drive with the letter Y: (which you probably don't have). Change the letter in all commands accordingly.

1. Obtain the package `Xbase` and install it from the root directory of the Y: drive, by entering the following commands:

```
[C:\] Y:
[Y:\] cd \
[Y:\] unzip \path_of_package\Xbase.zip
```

2. Edit your CONFIG.SYS file to contain the following line somewhere:

```
DEVICE=Y:\XFree86\lib\xf86sup.sys
```

   Of course replace ``Y:'' with the correct drive letter.

3. At this point, you may consider to add the variables required for XFree86/OS2 as well, which will save you from one additional reboot. Refer to section Adding Variables to CONFIG.SYS below.
4. After adding the device driver entry to the CONFIG.SYS file, you must reboot to install the driver. XFree86/OS2 will not work without this driver.
5. Start a full screen OS/2 CMD session and enter the following command:

```
[C:\] Y:\XFree86\bin\SuperProbe
```

6. This command will (normally) report important information about your video configuration, i.e. the type of chipset, the available video memory and the RAMDAC circuit available. Please write this down or redirect the output of ``SuperProbe'' into a file by entering:

```
[C:\] Y:\XFree86\bin\SuperProbe >filename
```

7. SuperProbe can identify many more video cards than are supported by XFree86. In some cases, SuperProbe unfortunately detects a wrong card, often it claims to have seen a MCGA card which is some sort of a fallback. Generally, if it is approximately right, there are only few reasons for doubts; if it is totally off (e.g. saying it has seen a ET4000, and you have a Cirrus card), you should report a mis-detection as a bug to the given address. In all cases, please take the few minutes and check the accompanying README.* files to check for special precautions, options, or features of the card.
8. If the README files tell you that your hardware is supported, please obtain the rest of the software.

Previous: *Troubleshooting*
Next: *Installing the packages*

TOC

8. Installing the packages

XFree86/OS2 assumes a directory hierarchy starting from `drive:\XFree86`. This can be changed, but is strictly discouraged.

1. Choose a HPFS partition with sufficient free space.
2. For each package to install, go to the root directory of this drive, and type:

```
drive:> cd \
drive:> unzip \path_of_packages\Xxxxx.zip
```

3. You might encounter that some packages report duplicate files, e.g. the X server packages install corresponding README files, which are also in the Xdoc package. This is okay, the files are the same. Let unzip replace the files.
4. No special sequence to unpack the files is required.

9. Adding Variables to CONFIG.SYS

XFree86/OS2 requires a number of settings in the CONFIG.SYS file to work correctly. Please add the following settings, and in particular take care to set forward versus backward slashes correctly:

TERM

Set the preferred terminal type for the xterm or editor to be used. Some programs need this setting. I have my type set to

```
SET TERM=ansi
```

`\XFree86\lib\X11\etc\termcap.x11` contains a suitable termcap which can be used in place of termcap files that come with EMX, EMACS, or other ported software.

TERMCAP

This variable must be set to the location where the termcap file used for the above `TERM` variable is searched. My setting, for instance, is:

```
SET TERMCAP=D:/EMX/ETC/TERMCAP.X11
```

Note that forward ``/'' is used as a directory separator.

ETC

Set to an ETC directory. Normally, this is already set to the ETC directory of the TCP/IP code, such as

```
SET ETC=C:\TCPIP\ETC
```

TMP

Set to an TMP directory. Normally, this is already set to the TMP directory of the TCP/IP code, such as

```
SET TMP=C:\TCPIP\TMP
```

HOSTNAME

Set to the internet hostname. Normally, this is already set by the TCP/IP installation program, such as

```
SET HOSTNAME=myhost
```

With IAK, you would normally run a loopback configuration Network configuration and would


TOC

then set this to

```
SET HOSTNAME=localhost
```

USER

LOGNAME

Set both to a username. Currently, they are there just to make some programs happy; in the future, this variable might be set by a login shell of a multiuser configuration. My variable, for instance, is set to

```
SET USER=holger
SET LOGNAME=holger
```

HOME

Set this to an existing directory that is supposed to be a home directory of a user. Some utilities place temporary and init files here. This is also future investment for a multiuser configuration, but must still be there. For instance, this variable might be set to

```
SET HOME=H:\user\holger
```

X11ROOT

This is one of the most important settings, it determines the root of the XFree86 directory tree. Normally, you will set this to the drive letter of the partition where the \XFree86 tree resides, such as in

```
SET X11ROOT=Y:
```

You may try to move the tree to another subdirectory, e.g. to `K:\OS2\X11\XFree86...` and would then have to change this to

```
SET X11ROOT=K:/OS2/X11
```

, but this is discouraged, since some utilities might not accept this. Note the forward ``/'' as a directory separator here.

DISPLAY

This variable may be set to the display to be used for displaying clients. Normally you will set this variable to the same value as the `HOSTNAME` variable and simply add a `:0.0` after it, such as

```
SET DISPLAY=myhost:0.0
```

Read the X11 man page on the exact meaning of these postfixes and other options.

XSERVER

Set this to the executable name of the X server to be used. This must be a complete path. My setting is as follows:


```
SET XSERVER=D:/XFree86/bin/XF86_Mach64.exe
```

PATH

Add the binary directory for the X11 utilities to your search PATH. This is normally the directory (adjust the letter)

```
Y:\XFree86\bin
```

It is possible to move the binaries to another directory in the search path; for maintenance reasons and clarity of the structure, this is not recommended, though.

LIBPATH

Add the DLL directory for the X11 utilities to the LIBPATH. This is normally the directory (adjust the letter)

```
Y:\XFree86\lib
```

It is possible to move the DLLs to another directory in the library path; for maintenance reasons and clarity of the structure, this is not recommended, though. Note that `Y:\XFree86\lib` has several other subdirectories; these may not be moved elsewhere, rather they must stay there, because most utilities form a path to these directories by using `%X11ROOT%\XFree86\lib` as a base.

The recent version of XFree86/OS2 has a REXX script named `checkinstall.cmd` which you can (and should) use to check whether you have entered most things correctly. This is not bullet-proof, but prevents the most obvious setup problems. Also, the X server itself will do some checking and will refuse to start if something is wrong.

README for XFree86 on OS/2 **:** *Adding Variables to CONFIG.SYS*
Previous: *Installing the packages*
Next: *Remarks on the Network Configuration*

10. Remarks on the Network Configuration

It is beyond the scope of this document to even give an introduction about the correct installation of the TCP/IP networking system. You must do this yourself or seek assistance elsewhere. It is only possible to say here that a PC working well in a TCP/IP based LAN network will also work with XFree86/OS2 (when all other prerequisites are matched as well).

With IAK, there is a special configuration necessary, unless you want to use XFree86/OS2 only during a hot link to your Internet provider, the so called ``localhost'' or ``loopback'' configuration. This is a local network interface which ``loops'' back to the same host. The following settings are necessary for this:

1. Create a file `\tcpip\etc\hosts` with the following content:

   ```
   127.0.0.1  localhost
   ```

2. Add the following line to your `\tcpip\bin\tcpstart.cmd`:

   ```
   ifconfig lo 127.0.0.1 up
   ```

   If you don't have such a tcpstart.cmd file (Warp 4 calls this file `\MPTN\BIN\MPTSTART.CMD`), create one, and add a line like the following to your config.sys file: `CALL=C:\OS2\CMD.EXE /Q /C C:\tcpip\bin\tcpstart.cmd >NUL`: (implying that your bootdrive is C:).

3. Set the HOSTNAME environment variable to `localhost` as described in the last section.

4. Add the following line to CONFIG.SYS:

   ```
   SET USE_HOSTS_FIRST=1
   ```

5. After rebooting, verify that the following command works:

   ```
   [C:\] ping localhost
   ```

You don't need this ``loopback'' interface if your PC is connected to a LAN (either directly or through SLIP/PPP).

In case of a SLIP/PPP line, you have to establish this connection BEFORE you start XFree86.

The `checkinstall.cmd` script coming with XFree86/OS2 gives some advice on the configuration as well.

If you have problems to get this or other basic networking things running, seek assistance elsewhere.

Next: *Configuring X for Your Hardware*

TOC

11. Configuring X for Your Hardware

After you have added the required settings and setup a working network, run the `xf86config` program to create a standard configuration file in `Y:\XFree86\lib\X11\XF86Config` from a windowed or full screen OS/2 text session:

```
[C:\] xf86config
```

The `xf86config` program will ask a number of questions. You will need the information obtained from the SuperProbe program here. The program should be self explanatory; if you have problems to understand something though, seek assistance in the newsgroups.

It is possible, but strongly discouraged for the non-expert, to edit the `XF86Config` file with a text editor. In a few situations as described in the FAQ, however, this might even be mandatory. This file is not a hacker's area, such as the Win95 registry, but it has in common with it that you can easily cause damage.

For details about the `XF86Config` file format, refer to the *XF86Config(4/5)* manual page.

If you know the configuration process from Linux or other XFree86 platform, you will encounter a few differences:

- There is no configuration for the mouse type or device. The mouse device name is fixed to OSMOUSE, and this cannot be changed. If you have a three-button-mouse, install the correct OS/2 driver for it, such as

  ```
  DEVICE=D:\OS2\BOOT\PCLOGIC.SYS SERIAL=COM1
  DEVICE=D:\OS2\BOOT\MOUSE.SYS TYPE=PCLOGIC$
  ```

  for a MouseSystems compatible mouse, for instance.
- The X server does not read the native OS/2 keyboard map, but the new XKB server extension might already give you a correct keyboard layout, provided your language was selectable in the `xf86config` program. If you encounter incorrect settings, please send a mail to `XFree86@XFree86.org` describing in detail what is wrong. Even with XKB, you have the option to replace some key settings with a xmodmap file. See the man page for xmodmap for details (or use some available xmodmap file from Linux - they are the same).
- There is no support for the Wacom and Elographics input devices yet.

In most cases, an existing `XF86Config` file for the same XFree86 version from Linux or another platform may be used without changes. There is one prominent exception: some S3 805 based VLB cards put their video memory in odd locations. The X server can search for this memory by experimentally mapping and unmapping possible memory regions. In XFree86/OS2, the OS may run out of memory

tiles during this process. If this happens, you must find out the location of the memory yourself and a
as an option

```
MemBase 0x12345678
```

to the XF86Config file.

Once you've set up a XF86Config file, you can fine tune the video modes with the `xvidtune` utility.

README for XFree86 on OS/2 **:** *Configuring X for Your Hardware*
Previous: *Remarks on the Network Configuration*
Next: *Running X*

12. Running X

16mb of memory is a recommended minimum for running the network software, X and the presentation manager in parallel. The server, window manager and an xterm take about 4-6 Mb of memory themselves. X will start up on a system with 8MB or less, but the performance will severely suffer from heavy disk swapping. Your mileage may vary, though, so some people might consider this still tolerable.

The easiest way for new users to start X windows is to type:

```
[C:\] startx
```

.

To get out of X windows, type: ``exit" in the console xterm. You can customize your X by creating `.xinitrc`, `.xserverrc`, and `.twmrc` files in the directory that the `HOME` environment variable points to. These files are described in the xinit and startx man pages.

By default, the systemwide `xinitrc` file (in `Y:/XFree86/lib/X11/xinit/xinitrc.cmd`) installs the rather simplistic `twm` window manager. You can find better window managers on the ported software page at http://set.gmd.de/~veit/os2/xf86ported.html .

13. Rebuilding the XFree86 Distribution

Do you really want to rebuild XFree86/OS2 from source? Read the file OS2.Notes on details to recompile XFree86/OS2 from scratch.

README for XFree86 on OS/2 **:** *Rebuilding the XFree86 Distribution*
Previous: *Running X*
Next: *Building New X Clients*

14. Building New X Clients

The easiest way to build a new client (X application) is to use `xmkmf` if an `Imakefile` is included in the sources. Type ``xmkmf -a'' to create the Makefiles, check the configuration if necessary and type ``xmake''. ``xmake'' is a wrapper for the GNU make program which defeats the improper SHELL setting typically found in a Makefile generated from an Imakefile. Also see the XFree86/OS2 FAQ for more hints about porting X clients.

TOC

15. Acknowledgements

Many thanks to:

- **Sebastien Marineau** for his great work on getting the server code debugged
- **Eberhard Mattes** for the wonderful base platform EMX which this port heavily relies on
- **ME** - no, no, forget this: I won't praise myself :-)

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/OS2.sgml,v 3.9.2.10 1999/08/02
08:38:16 hohndel Exp $
```

```
$XConsortium: OS2.sgml /main/4 1996/03/11 10:46:06 kaleb $
```

Notes on Rebuilding XFree86/OS2 from Scratch

Holger Veit

Last modified August 1st, 1999

1. Preface

2. Tools required

3. Compiling and Installing

1. Preface

X11 and XFree86 were initially developed on Unix-based systems. Usually Unix systems provide a rich number of tools and utilities to get certain things done. Under OS/2, these tools are not installed, but ports are available which are sometimes functionally equivalent to Unix utilities with the same name, but also differ sometimes in a subtle way. This guide will give you hints if you intend to rebuild the system from scratch under OS/2.

Please also read README.OS2 for end-user information, and set at least the environment variables described there.

At the current time, the most recent version available is XFree86-3.3.5. This is a full and unrestricted version which comes with complete source code. 3.3.5 is not only a bugfix release, but also supports new hardware, some of which might not even supported by OS/2 itself. See the RELEASE NOTES document for details.

If you want to join the XFree86 developer team, e.g. to add support for certain hardware, please send a request to BOD@XFree86.org. Please think about such a step carefully before, though, since much work is involved. Please use the XFree86-3.3.5 source code as a test example how to compile the system. The ability to manage that is a basic requirement for becoming a developer.

2. Tools required

I have tried to reduce the number of external tools, but when looking back it seems I were not very successful. At least I managed to get everything working with the native CMD.EXE shell only. However, there is still plenty of software required. Most of this software is available from `hobbes.nmsu.edu` or `ftp.leo.org` via anonymous FTP. The following shopping list shows what you will need:

- gcc EMX/gcc emx 0.9C patch4 or later (0.9d preferred!)
- gzip GNU zip/unzip
- tar GNU tar
- patch Larry Wall's patch utility (attention: incompatible tool with same name in OS/2)
- install BSD/GNU install
- rm,mv,cp GNU file utilities
- tee,.. GNU shell utilities
- groff GNU nroff/troff
- sed GNU sed stream editor
- grep GNU grep
- gawk GNU awk
- make GNU make 3.71/3.72 (use the one from Xprog.zip!)
- flex GNU flex
- bison GNU bison
- find GNU find (attention: incompatible tool with the same name in OS/2)

If there is no version number given, any new version will do. Particularly critical is only EMX/gcc and GNU make. Note that the second GCC implementation which might still be available from some archives is NOT compatible.

Furthermore, you need the XFree86 sources. These are available from the common XFree86 repositories. Look into a directory which is often named /pub/XFree86/3.3.5/source.

3. Compiling and Installing

You need about 300MB of free HPFS space for the whole system. This does not include space for the postscript and troff documentation files. I have never installed them. Nor did I install the test subtree.

1. Install all the above utilities. Refer to the corresponding documentation. Verify that everything works well, particularly EMX.
2. It is a good idea to use the same or a similar structure I have. I have made a directory `\x11` on the partition for compiling and have put everything below this tree. I found that a clean tree occupies less than the half space of the disk, this gives me the opportunity to rename this tree to `\x11old` and copy a new version to the same disk to produce diffs. Last time the complete tree was arranged under the root directory `xc`, this would become `\x11\xc` then.
3. To unpack the files you would usually execute the command

   ```
   gzip -dc file.tar.gz | tar xvf -
   ```

   in the `\x11` directory. At the end you will usually see the irritating, but non-fatal message "gzip: stdout Broken pipe". Ignore it.
4. After that, is is likely necessary to apply some patches, either from the XConsortium or from the XFree86 project. Before you do this, enter

   ```
   chmod -R a+rw \x11\xc
   ```

   to make certain files in the tree writable.
5. There should be a file `added-XXX` accompanying the patch file which lists the files that are newly created. The patch program has a problem with creating new directories, so we need to create them on advance. For each `added-XXX` file you find, execute from `\x11`

   ```
   xc\config\util\added added-XXX
   ```

   If there is no `added-XXX` file available, you can make one with the following instructions:

   ```
   grep "\*\*\* xc/" patchfile >added-file
   ```

   Edit `added-file` with a text editor and remove the `***` at the beginning and the time stamp at the end (search for a TAB and erase to the end of the line). You get a list of file paths, one in a line, which is the input to the added utility.
6. After that you can apply the patches in the right order. Usually this is done by a command

   ```
   patch -p -E <patchfile 2>&1 | tee patchlog
   ```

   from the `\x11` directory. Be aware to use the right patch - OS/2 has a utility with the same name and different functionality. Don't use the recommended `-s` option, this makes `patch` quiet, and you won't see problems in the patchlog file. Use

   ```
   find \x11 -name *.rej -print
   find \x11 -name *# -print
   ```

to find any rejects and unapplied patches (attention: yet another OS/2 program with wrong functionality). Normally there shouldn't be any problems of this kind, else you have made a mistake. Finally remove the original files with

```
find \x11 -name *.orig -print -exec rm {} ;
```

7. Go to the `xc/config/cf` directory and edit the `xf86site.def` file to match your requirements (you probably don't want to compile all X servers). Certain changes must be set to the following values:
   - Disable if not already done any PC98 server; PC98 (Japanese XFree86) does not work yet. Porters from Japan are welcome!
   - `#define WacomSupport NO #define ElographicsSupport NO` Both options are not yet supported.
   - Tcl* and Tk* don't need to be set explicitly. Reasonable defaults are in the other config files, provided you have a complete XFree86/OS2 binary tree with the tcl/tk runtime support installed.
   - `#define BuildDynamicLoading NO` This does not work.
8. Go to the directory `xc\util\compress` and `make compress.exe` there. Install the program produced there in your path. I stumbled more than once on half-ported compress programs on OS/2 ftp servers that are defective w.r.t. reading and writing stdin/stdout. In some stage (font compression) otherwise you will get a core dump of mkfontdir, because all compressed fonts are corrupt.
9. Set the environment variable `X11ROOT` to something different than it is; otherwise the installation process will overwrite your original XFree86/OS2 installation. If you have not set this variable, go back to the prefix section of this document: you have forgotten something.
10. Copy the file `xc/programs/Xserver/hw/xfree86/etc/bindist/OS2/host.def.os2` to the location `xc/config/cf/host.def`. Use this file to do any specific modifications to imake variables, rather than editing the file xfree86.cf, imake.tmpl, or os2.cf directly.
11. Copy the file `xc/config/util/buildos2.cmd` into the `xc` directory. If this is a second or later attempt, you might need to copy the saved toplevel Makefile.os2 back to Makefile.
12. Execute this `buildos2.cmd` command in the `xc` directory; it will produce a logfile `buildxc.log` in this directory.
13. Go have a bucket of coffee, or better, buy new coffee - in Colombia! The compile will need between 2 and 20 hours, depending on your selections, and the horse power of your hardware.
14. When finished, view the logfile for errors, and fix the problems if there are some. I have managed to compile the whole system flawlessly, so there is at least one configuration that works.
15. Finally, from the `xc` dir, execute

    ```
    xmake install
    xmake install.man
    ```

16. There are a few minor glitches in the installation:
    1. The xdm and linkkit directories will fail in compile and installation. This is no problem and has no effect on the rest of the system.
    2. The imake.exe which is installed in `\XFree86\bin` is usually defective. The one which was built initially and installed in the root directory of the drive where you have the source tree is okay. So simply copy this `\imake.exe` to the `\XFree86\bin` directory manually. Some day this might be fixed.
    3. `XF86Setup` is not ported yet and won't work with the tcl/tk port available for XFree86/OS2. My idea was to replace this by some native installation tool, which I didn't find the time to do yet. Feel free to spend a bit of time to play with XF86Setup if you like.

Well, you see, this was quite easy :-)


```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/OS2note.sgml,v 3.4.2.6 1999/0
08:38:17 hohndel Exp $
```

```
$XConsortium: OS2note.sgml /main/1 1996/02/24 10:08:59 kaleb $
```

Notes on Rebuilding XFree86/OS2 from Scratch **:** *Compiling and Installing*
Previous: *Tools required*
Next: *Notes on Rebuilding XFree86/OS2 from Scratch*

Information for SCO Users

J. Kean Johnston (hug@netcom.com)

30 November 1996

1. Binary Distribution
2. Source Distribution
3. Before Running XFree86
4. Switching Consoles
5. Setting up Man Pages
6. Using SCO binaries/servers.
7. Compiling XFree86 under Open Server 5
8. Relevant Documentation
9. Known Problems
10. Trouble Shooting
11. Acknowledgements

1. Binary Distribution

The following files are provided in the binary distribution:

README.SCO

This file.

gunzip.Z

The GNU `uncompress` utility.

***X312Xdoc.tgz**

The XFree86 specific documentation.

X312Mono.tgz

The Mono server

X312VG16.tgz

The 16 colour VGA server

X312SVGA.tgz

The Super VGA server

X312S3.tgz

The S3 server

X3128514.tgz

The 8514 server

X312AGX.tgz

The AGX server

X312Mc32.tgz

The Mach 32 server

X312Mc64.tgz

The Mach 64 server

X312Mc8.tgz

The Mach 8 server

X312P9k.tgz

The P9000 server

X312W32.tgz

The ET4000/W32 server

***X312cfg.tgz**

The local configuration files for `xdm/fs/xinit`.

***X312bin.tgz**

The `bin` directory, contains most executables.

***X312lib.tgz**

The shared and unshared libraries.

***X312fnt1.tgz**

`75dpi` and `misc` fonts.

X312fnt2.tgz

`100dpi` and `Speedo` fonts.

***X312inc.tgz**

The X11 include files.

X312man.tgz

The formatted man pages.

X312lkit.tgz

The server link kit (all drivers + PEX).

X312util.tgz

Lots of PD utilities provided as is.

X312pex.tgz

All files relating to PEX including libraries and header files. The LinkKit is required to obtain servers capable of running PEX.

To obtain a minimum XFree86 installation you will require the archives marked with a `*' above, the server binary best suited to your machine and optionally "`gunzip.Z`". All the files are compressed with "`gzip`" except of course "`gunzip.Z`" which is compressed using the conventional `compress` program.

To install the XFree86 binaries just follow these steps.

1. Obtain the files you require.

   The rest of this procedure must be done as root. If you do not run the extraction as root the permissions on the files will not be correct. For example, the `X' server is s-bit root and will not function correctly if extracted as an ordinary user.

2. create a directory `/usr/X11R6`, permissions `755` should do nicely.
3. cd `/usr/X11R6`
4. extract the archives, for example:

```
gunzip < X312bin.tgz | tar xvpf -
```

5. if you have installed man pages see the later section on setting up man pages.
6. Look through `/usr/X11R6/lib/X11/doc/INSTALL`, especially section 2 on configuring and using XFree86. This should allow you to get a server up and running. Before starting the server check in the later section Before Running XFree86, in this document, to see if there are any system requirements you have to make for the server to operate correctly.

Information for SCO Users **:** *Binary Distribution*
Previous: *Information for SCO Users*
Next: *Source Distribution*

2. Source Distribution

The SCO port comes as part of the standard XFree86 distribution. Consult the XFree86 `README` for more information on the location of sources.

Please note that as of XFree86 3.2, Only SCO Open Server Release 5 and onwards are supported. If you are using a previous version of SCO UNIX and you want to use XFree86, use the 3.1 series, or be prepared for build failures.

For people who want and need to look around the source, there are now two files in ``xc/config/cf". Firstly, ``sco.cf" is the old original SCO configuration file, and ``sco5.cf", which is the currently used configuration file.

3. Before Running XFree86

The SCO `xterm` terminfo description is not compatible with the `xterm` in the R5 distribution.

To use a Bus/Keyboard or PS2 mouse you should configure the mouse drivers under SCO as above using '`mkdev mouse`'. You may then use the `OsMouse` option in your XF86Config to specify that XFree86 should use the SCO mouse drivers. To do this, set the `Protocol` to "`OsMouse`" in the Pointer section of your XF86Config file. You can also use "`OsMouse`" for your serial mouse, especially if you are having trouble getting your mouse to work using the XFree86 mouse drivers.

If you do not have the SCO TCP/IP package installed do not panic. XFree86 will work fine without TCP/IP but you will most likely have to do some or all of these things:

- Do not worry about errors from the X server complaining about ``/dev/socksys''. The X server is configured to run on systems with and without TCP/IP. This error is just pointing out that you do not have TCP/IP and that this method of connecting to the server has been disabled.
- Do worry about errors involving ``/dev/spx'' or the ``sco'' connection type. This means something is wrong with the streams pipes that are used for connections on the local machine. First be sure that your server has the ``s-bit'' set. You can do this by running this command for the X server you are using: ls -al /usr/X11R6/bin/XF86_XXXXXX The output should contain the `s' character instead of the `x' character. For example:

  ```
  -rwsr-xr-x   1 root     bin      1074060 Jul 24 11:54 XF86_W32
  ```

  is correct while:

  ```
  -rwxr-xr-x   1 root     bin      1074060 Jul 24 11:54 XF86_W32
  ```

  is not.
- you may have to install streams into the kernel with ``mkdev streams'' Check the SCO Manuals for more information on this.
- you may have to configure some devices in /dev, check in the "Trouble Shooting" section of this document for the entry which comments on ``/dev/spx'' and ``Xsco''.
- Your streams resources may be configured too low. You should check your streams parameters against the following values, if the are higher then you do not need to changes them. To check these values, login as root, change directory to ``/etc/conf/cf.d'' and then run ``./configure''. Once you are running configure, choose the ``Streams Data'' option and step through the entries. Just press <ENTER> at each prompt unless you want to change a value. The values to look for, and their minimum values, are:


TOC

```
NSTREAM      128
NQUEUE       512
NBLK4096     4
NBLK2048     32
NBLK1024     32
NBLK512      32
NBLK256      64
NBLK128      256
NBLK64       256
NBLK16       256
NBLK4        128
NUMSP        128
```

You will not normally need to change any of these, if however you do have to change some, configure will confirm that you want to save the changes before exiting, and will give you further instructions on rebuilding the unix kernel.

Information for SCO Users **:** *Before Running XFree86*
Previous: *Source Distribution*
Next: *Switching Consoles*

4. Switching Consoles

XFree86 uses similar console switching keys as the SCO R4 and R5 servers. That is, `Ctrl-PrntScr` takes you to the next console along from the one X is running on. If this is the last console it will take you to console 1. `Ctrl-Alt-FXX`, where `XX` is a function key between `F1` and `F12` will switch you to the console number assigned to that function key. `F1` corresponds to `tty01` (or console 1), `F2` corresponds to `tty02` (or console 2) etc. Those interested in modifying the console switching should look in `xc/programs/Xserver/hw/xfree86/common/xf86Events.c`.

5. Setting up Man Pages

After compiling the tree, or after installing the binary distribution you can get man to recognise the XFree86 man pages by adding `/usr/X11R6/man` to the `MANPATH` in `/etc/default/man`, the line should look similar to:

```
MANPATH=/usr/man:/usr/X11R6/man
```

This allows all users to view the X man pages. You may change your own `MANPATH` environment variable if you do not want everyone to access the man pages.

By default the man pages are compressed using ``compress'' to conserve space. If you do not want to compress the man pages change `CompressManPages` to `NO` in your ``xf86site.def'' file. Those using the binary distribution can use ``uncompress'' to uncompress the man pages.

6. Using SCO binaries/servers.

XFree86 will accept connections from SCO binaries (R3 upwards) and the SCO R5 server will also accept connections from XFree86 binaries. This means you may mix and match the two if you have ODT. For example you may still use the Motif window manager (mwm) if you prefer.

7. Compiling XFree86 under Open Server 5

As of GCC version 2.8.0, Open Server is supported. Configure it by using the following:

```
./configure i486-sco3.2v5.0
```

There is no reason to modify gcc in any way. It compiles cleanly on Open Server 5.

SCO Open Server 5.0 is recognised automatically by XFree86. You do not need to specify any `BOOTSTRAPCFLAGS` parameters when doing a `make World`. You can ignore the warning message about `BOOTSTRAPCFLAGS` at the very beginning of a `make World`.

1. Fine tune ``site.def/xf86site.def"

   Use GCC if you can. XFree should compile with the DevSys cc, but GCC has better optimizations, and is guaranteed to work.
2. SCO Open Server comes with Visual TCL, which is an old (and incompatible) version of TCL. If you want to use XF86Setup you will have to compile Tcl and Tk yourself. Both are supported well on SCO Open Server 5. Tcl 7.6 and Tk 4.2 are available from `ftp://ftp.smli.com/pub/tcl`.
3. You may want to disable dynamic loading support. Several users have reported trouble with this. XIE and PEX5 definitely do not work. If you want to experiment, try enabling this. Please report successes or failures to me.
4. Do **not** enable the `HasSVR3mmapDrv` as you may have done in older versions of SCO. Open Server 5 has full mmap() support, and this is used for direct frame buffer access.
5. If you know you will not ever be using COFF binaries, and you are short of space, set `ForceNormalLib` to `NO`. Doing this will cause only the ELF versions of the libraries to be built. ``sco5.cf" sets this to `YES` by default, so you must explicitly set it to `NO` in ``xf86site.def". All binaries are compiled in ELF mode to reduce space.

8. Relevant Documentation

Some relevant documentation for SCO Users and Developers can be found in the following files.

README

the standard XFree86 `README` (`/usr/X11R6/lib/X11/doc`)

README.SVR3

Although a lot of this readme is based on Interactive a substantial proportion is still relevant.

All of the VGA/Config documentation.

`/usr/X11R6/lib/X11/doc/VideoModes.doc` and the `README` files for particular video cards.

9. Known Problems

- After running the server you may see some strange characters in your input to the shell. This is due to some unprocessed scancodes and is of no concern. This will be fixed in a future release.
- Not all of the applications in `/usr/X11R6/bin` have been debugged.

10. Trouble Shooting

Problem:

The server does not start up, and I cannot tell what is going wrong as it did not print any error messages.

Causes:

There can be any number of causes why the server doesn't start. The first step is to find out what the server has to say. To do this we have to catch the error output of the server into a file. This output contains a log of what the server is finding/doing as it starts up. To get this output run:

```
startx 2> /tmp/errs
```

The output of the server will now be in "/tmp/errs". You should look through this output for possible problems, and then check here in this document for any references to the problems you are seeing.

Problem:

The server starts up, the screen goes blank, and I never see anything else. It appears that my machine has hung.

Causes:

Again this can have many causes. Most likely your XF86Config is wrong. You should be able to kill the server by typing Ctrl-Alt-BackSpace, if it is still running. If this does not restore your display then you may have to drive your system blind. Always keep another login running at the shell prompt so that you may switch to that screen and run commands even if you cannot see anything on the screen. Try these things, usually in the order given:

- log out of the login where you started ``X" and then change consoles. This will cause the SCO screen switching code to try to reset the card.
- run ``vidi v80x25", this command will also try to set your card into a viewable mode.
- shutdown the machine cleanly with ``shutdown" and try again.

When first trying to get XFree86 to run, be sure to use a simple setup. Get 640x480 working first then move on to higher resolutions. Always trap the output of the server as shown earlier. Once you have the valid clocks for your video card (as provided in the server output), hard code them into your XF86Config as this will take some strain off your monitor during XFree86 startup where it usually probes the various clock frequencies. Getting the ``X" screen to appear can be a painfully slow task. Be patient

and read as much of the doco as you can handle. You will get it to work.

Problem:

```
Fatal server error:
xf86MapVidMem:No class map defined for (XXXXX,XXXXX)
```

Causes:

1. Your system does not have the correct /etc/conf/pack.d/cn/class.h, You can confirm this by editing the file and looking for the string "SVGA", if it is not there then you should re-install this file from the "Extended Utilities" diskettes provided with your OS. If this is not possible then installing the "dmmap" driver from the distribution may allow the server to operate correctly.

Problem:

`xf86install` does not work.

Causes:

You should not be running `xf86install` when using the XFree86 server under SCO. It is used for Interactive (ISC) installations.

Problem:

The server starts but the screen is not aligned correctly or is shaky and impossible to view.

Causes:

This is most likely due to an incorrect `XF86Config` setup. Look for the files `README.Config` `VideoModes.doc` (in `/usr/X11R6/lib/X11/doc` with the binary distribution). These files explains how to fix up your video modes.

Problem:

1. Can only run a limited number of xterms.
2. `xterm` does not work but other programs like `xclock` do work.

Causes:

Not enough or no pseudo ttys devices are present on your system. Run "`mkdev ptty`" and increase the number of ptty's.

Problem:

When running curses/termcap applications in an `xterm` the output gets corrupted especially when scrolling.

Causes:

1. You are running an original 1.3 distribution of XFree86. Update to the latest version (3.2 or greater).
2. You have resized the window and not ran "`eval `resize`" before using your application. The SCO operating system does not support dynamic resizing of xterms fully so this command must be run after resizing an `xterm` in order for curses/termcap applications to operate correctly.

Problem:


TOC

1. When starting X it dies with an error "Cannot access a needed shared library".
2. When starting an X application is dies with the above error.

Causes:

1. You do not have the binaries installed in the correct directory. Check that they are in `/usr/X11R6`
2. You have upgraded to a new binary distribution which has a new version of the shared libraries which are not compatible with your old binaries. To fix this you will need to re-install the old shared libraries or recompile your application against the new libraries.

Problem:

When linking against the SCO motif library I get an unresolved external for "`XtDisplayStringConversionWarning`" when using gcc.

Causes:

The SCO library is compiled with limited length identifiers. To work around this add the following code to your application when compiling under XFree86 with gcc and SCO motif.

```
#ifdef SCO
void XtDisplayStringConversionWarnin(dpy, from, toType)
    Display*  dpy;
    String    from;
    String    toType;
{ XtDisplayStringConversionWarning(dpy, from, toType); }
#endif
```

Problem:

The server fails to run and prints out a line similar to:

```
XFree86: Cannot open /dev/spx for ???? listener: No such
file or directory
```

Causes:

All SCO unix installations appear to have the Streams pseudo tty driver installed, but not all the devices are present.

1. there should be a `/etc/conf/pack.d/sp` directory,
2. `/etc/conf/sdevice.d/sp` should have a 'Y' in it.
3. You need a file in `/etc/conf/node.d` which contains something like:

```
clone   spx             c       sp
sp      X0S             c       127
sp      X0R             c       126
sp      X1S             c       125
sp      X1R             c       124
sp      X2S             c       123
sp      X2R             c       122
```


```
sp      X3S             c       121
sp      X3R             c       120
sp      X4S             c       119
sp      X4R             c       118
sp      X5S             c       117
sp      X5R             c       116
sp      X6S             c       115
sp      X6R             c       114
sp      X7S             c       113
sp      X7R             c       112
```

if you don't have something like this (maybe called "Xsco") then create one and that should fix your problem. As far as I can tell the streams pseudo tty driver should be there.

The simplest way to get the devices if you had to create this file is to rebuild the kernel and the environment. If you don't want to do this then:

```
touch /etc/.new_unix
cd /etc/conf/bin
./idmkenv
```

and try it out.

Information for SCO Users **:** *Trouble Shooting*
Previous: *Known Problems*
Next: *Acknowledgements*

11. Acknowledgements

Thanks to the Core team for their previous and continuing help with the SCO work. Many thanks to **Stacey Campbell** at SCO for all the advice and insights provided. Thanks to SCO in general for making information available for XFree86 development.

Thanks also to **Peter Eubert** (*peter.eubert@iwb.mw.tu-muenchen.dbp.de*) and **Kent Hamilton** (*kenth@stl.scscom.COM*) for input on compiling under 3.2.4 systems. **Larry Plona** (*faxi@world.std.com*) and **Didier Poirot** (*dp@chorus.fr*) for their input on xdm and 3.2.4 compilation under 3.1. And of course the beta list for its input on everything.

Special thanks to **Jerry Whelan** (*guru@stasi.bradley.edu*) for providing an ftp site for the binary distribution.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/SCO.sgml,v 3.16 1997/01/25 03:22:12
dawes Exp $
```

```
$XConsortium: SCO.sgml /main/11 1996/10/23 11:45:55 kaleb $
```

Information for Solaris for x86 Users

David Holland

25 Feb 1998

1. What is XFree86
2. Solaris for x86, versions on which XFree86 3.3.3 has been tested
3. The VT-switching sub-system in Solaris x86
4. Notes for building XFree86 on Solaris x86
5. Notes for running XFree86 on Solaris x86
6. Known bugs, and work arounds with Solaris x86
7. Bug Notification

1. What is XFree86

XFree86 is a port of X11R6.3 that supports several versions of Intel-based Unix. It is derived from X386 1.2 which was the X server distributed with X11R5. This release consists of many new features and performance improvements as well as many bug fixes. The release is available as source patches against the X Consortium code, as well as binary distributions for many architectures.

The sources for XFree86 are available by anonymous ftp from:

ftp://ftp.XFree86.org/pub/XFree86/current

Solaris binaries for XFree86 are available for anonymous ftp from:

ftp://ftp.XFree86.org/pub/XFree86/current/binaries/Solaris

2. Solaris for x86, versions on which XFree86 3.3.3 has been tested

XFree86 3.3.2 has been actively tested on:

- Solaris 2.5.1 for x86
- Solaris 2.6 for x86

3. The VT-switching sub-system in Solaris x86

The virtual terminal sub-system is a undocumented, and unsupported feature of Solaris x86. Therefore if you use Virtual Terminals, you do so at **YOUR OWN RISK**.

The virtual terminals of Solaris work basically the same way as most other Intel based SVR4 VT sub-systems. However, there are a number of limitations documented below.

First, if you are running a Solaris 2.4 x86 system, and you want VT's, you will have to create the necessary devices first, so become root.

First verify the chanmux device driver's major number is 100:

```
# grep -i chanmux /etc/name_to_major
chanmux 100
#
```

If the number after 'chanmux' is anything but 100, I would suggest you immediately abort your attempt to create virtual terminals, and learn to live without them.

However, if it is 100, then as root type the following commands to create the maximum allowable number of virtual terminals.

```
# cd /dev
# mknod vt01 c 100 1
# mknod vt02 c 100 2
# mknod vt03 c 100 3
# mknod vt04 c 100 4
# mknod vt05 c 100 5
# mknod vt06 c 100 6
# mknod vt07 c 100 7
```

There is no need for a reconfiguration boot.

Secondly, for both 2.1, and 2.4 x86 systems, add a few lines to the `inittab` to enable logins on them.

(**Note**, do **NOT** make a mistake here, you could lock yourself out of the system)

```
---------------------->Snip Snip<----------------------------------------------
v1:234:respawn:/usr/lib/saf/ttymon -g -h -p "`uname -n` VT01 login: " -T AT386 -d
/dev/vt01 -l console
v2:234:respawn:/usr/lib/saf/ttymon -g -h -p "`uname -n` VT02 login: " -T AT386 -d
/dev/vt02 -l console
v3:234:respawn:/usr/lib/saf/ttymon -g -h -p "`uname -n` VT03 login: " -T AT386 -d
/dev/vt03 -l console
v4:234:respawn:/usr/lib/saf/ttymon -g -h -p "`uname -n` VT04 login: " -T AT386 -d
/dev/vt04 -l console
---------------------->End Here<------------------------------------------------
```

These four lines enable four VT's on Alt-SysReq-F1 through Alt-SysReq-F4.


TOC

Then execute the command '`init q`' to immediately enable the virtual terminals.

The keys used for VT switching are as follows:

`Alt-SysReq-F1` through `Alt-SysReq-F7` enable VT screens 1-7 respectively (if the VT is active).

`Alt-SysReq-n` enables the next active VT screen.

`Alt-SysReq-p` enables the previous active VT screen.

`Alt-SysReq-h` returns to the console.

If you are using virtual terminals, you must leave at least one free for use by the Xserver.

Limitations of the virtual terminal sub-system under Solaris x86:

There are only a total of 8 available VT's (7 normal VT's + 1 console) not the usual 15. If you have all 8 allocated, and you attempt to allocate a additional VT you will panic the system. (This bug is worked around in the Solaris XFree86 Xserver.)

From a programming stand point, they work pretty much as documented in the AT&T Unix System V/386 Release 4 Integrated Software Development Guide, however a number of `ioctl()` calls are broken.

Information for Solaris for x86 Users **:** *The VT-switching sub-system in Solaris x86*
Previous: *Solaris for x86, versions on which XFree86 3.3.3 has been tested*
Next: *Notes for building XFree86 on Solaris x86*

4. Notes for building XFree86 on Solaris x86

1. The majority of all modifications you will need to make are now in `~xc/config/cf/xf86site.def`.
2. Both Gcc, and ProWorks are supported by XFree86. Gcc-2.5.8 or gcc-2.7.2.3 are suggested, Gcc-2.6.0 is known not to work. You also need to set HasGcc2 correctly in `~xc/config/cf/xf86site.def`. You should also make certain your version of GCC predefines `sun'. 2.4.5 is known NOT to by default. If needed edit `/usr/local/lib/gcc-lib/*/*/specs`, and modify the `*predefines:` line.

   Note: A Threaded Xlib compiled with GCC has subtle problems. It'll work 98% of the time, however clients will occasionally exhibit strange hangs. Most notably image viewers such as xv-3.10 exhibit this problem.

   It is recommended that you set ThreadedX in `~xc/config/cf/sun.cf` to NO, if you are using GCC. ProWorks does not have this problem.
3. To build XFree86 with gcc you need gcc and (optionally) c++filt from GNU binutils. Don't install gas or ld from GNU binutils, use the one provided by Sun.

   With XFree86 3.3.2, you will need to setup a /opt/SUNWspro/bin directory containing symbolic links named `cc`, `CC`, and `c++filt` pointing respectively to the actual `gcc`, `g++` and `c++filt` commands.
4. If you don't have c++filt or if you have troubles in making World with c++filt, you need to set UseExportLists to NO in `~xc/config/cf/host.def`.
5. If you are using ProWorks to compile the XFree86 distribution, you need to modify your PATH appropriately so the ProWorks tools are available. Normally, they should be in `/opt/SUNWspro/bin`
6. You **MUST** put `/usr/ccs/bin` at the front of your PATH. There are known problems with some GNU replacements for the utilities found there. So the `/usr/ccs/bin` versions of these programs must be found before any other possible GNU versions. (Most notably GNU '`ar`' does not work during the build).
7. If you wish to use the "memory aperture" feature of the S3, and Mach32 servers, you need to compile, and install the Solaris x86 aperture driver for memory mapped I/O support. This driver is **REQUIRED** for the I128, P9000 and Mach 64 servers.

   You need to set HasSolx86apertureDrv to YES in `~xc/config/cf/xf86site.def`.

   to enable the aperture driver.

   Under Solaris 2.5 and later, there's a system driver (`/dev/xsvc` that provides this functionality. It will be detected automatically by the server, so you don't need to install the driver.

For Solaris 2.1 and 2.4, the source for this driver is included in `~xc/programs/Xserver/hw/xfree86/etc/apSolx86.shar`. Building, and installing the driver is relatively straight forward. Please read its accompanying README file.

5. Notes for running XFree86 on Solaris x86

1. If you have not made the Virtual Terminal devices, you will need to specify the terminal device to run the Xserver on. The correct device is vt00 so your `xinit` command would look like so:

   ```
   xinit -- vt00
   ```

   If you have made the virtual terminal devices you do not need to specify the VT to run the Xserver on.

   To be able to run XF86Setup, you must at least create /dev/vt01. Otherwise XF86Setup won't start.
2. For Solaris you will probably want to set your LD_LIBRARY_PATH to `/usr/X11R6/lib:/usr/openwin/lib:/usr/dt/lib`. Including `/usr/X11R6/lib` in your LD_LIBRARY_PATH is probably not necessary, however it doesn't hurt. :)

   Including `/usr/openwin/lib` in the LD_LIBRARY_PATH is recommended because some Sun supplied binaries were not compiled with LD_RUN_PATH set properly at compile time.

   Motif and CDE applications may require `/usr/dt/lib` in your LD_LIBRARY_PATH too.
3. Xqueue is **NOT** supported under Solaris. The includes necessary for Xqueue are available, however the driver does not seem to be in the kernel. (Go figure)
4. If you want to use xdm with Solaris, extract the files from the shar file in `/usr/X11R6/lib/X11/etc/XdmConf.svr4` into a temporary directory. The `README` file tells where the individual files need to be installed. Be sure to read through each file and make any site-specific changes that you need.

6. Known bugs, and work arounds with Solaris x86

1. The Solaris 2.1 for x86 OpenWindows filemgr does not work against a X11R5 Xserver, it probably will also not work against a X11R6 Xserver. Attempting to 'Drag and Drop' a file causes the filemgr to abort with a 'X error'

   Solaris 2.4 does not have this problem.

   There is no known work around.

TOC

7. Bug Notification

Bug reports need to be sent to **XFree86@XFree86.org**, or posted to the comp.windows.x.i386unix newsgroup. Questions or comments about the Solaris support, or the Solaris distribution need to be made to *davidh@use.com*, or *danson@lgc.com*.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/SOLX86.sgml,v 3.12.2.7 1998/11/07
13:37:49 dawes Exp $
```

```
$XConsortium: SOLX86.sgml /main/7 1996/10/28 05:43:28 kaleb $
```

Information for SVR4 Users

The XFree86 Project, Inc

27 Feb 1998

NOTE: *If you intend to use any of the accelerated servers, read section 10 and follow the instructions. Otherwise the X server will crash when exiting, restarting, or switching VTs.*

---

1. SVR4 versions on which XFree86 has been tested

2. How to cope with VT-switching hotkeys

3. Running SVR3 binaries on SVR4.0.4 and SVR4.2

4. Notes for building XFree86 on SVR4

5. Notes for running XFree86 on SVR4

6. Building non-core clients with SVR4

7. Using DOS/Merge 2.2 with XFree86

8. Keyboard mapping problems with some Esix systems

9. 106 Japanese keyboard problem on PANIX

10. A kernel patch that is required for accelerated servers

11. Other problems

1. SVR4 versions on which XFree86 has been tested

XFree86 has been tested on the following versions of **SVR4.0**:

- Microport: 2.2, 3.1, 4.1, 4.2
- Esix: 4.0.3A, 4.0.4, 4.0.4.1
- Dell: 2.1, 2.2, 2.2.1
- UHC: 2.0, 3.6
- Consensys: 1.2
- MST: 4.0.3
- AT&T: 2.1, 4.0
- ISC: 4.0.3
- NCR: MP-RAS
- PANIX: 5.0

and the following versions of **SVR4.2**:

- Consensys
- Novell/SCO UnixWare 1.x and 2.0

Basically, we believe that XFree86 binaries will run unmodified on any ISA, EISA, or MCA platform version version of SVR4.0 (Solaris 2.x is an exception), or SVR4.2. If you run XFree86 on another version of SVR4 that's not in this list, please let us know about it.

2. How to cope with VT-switching hotkeys

Some versions of SVR4 (Esix and Microport) have mechanisms for enabling two-key sequences for VT switching (`Alt-Fn`). The standard SVR4 mechanism is `Alt-SysReq-Fn`, which all versions we know use. Running under X, the `Alt-Fn` sequences are stolen by the driver before the server can see them, so you can't use them for X applications. So you want to switch back to the standard 3-key sequences while you are running X. Here's how to do it:

Microport

Microport makes this very simple. The 2-key mode is called "Microport Mode", and the 3-key mode is called "Compatible Mode". You enter Microport Mode by pressing `Alt-SysReq-m`. You enter Compatible Mode by pressing `Alt-SysReq-c`. So all you need to do is press `Alt-SysReq-c` after starting the X server to allow X clients access to the `Alt-Fn` sequences.

Esix

Esix has no keyboard-driven way to switch modes. There are two levels at which this can be handled:

1. There is a kernel tunable that determines which mode is the default. The tunable is the initialisation of kd_2keysw in `/etc/conf/pack.d/kd/space.c`. When set to 1 (the default), 2-key mode is enabled. When set to 0 it is disabled.
2. The mode can be changed for individual VTs programatically by an ioctl(). To make life easier for XFree86 users, a program called `2key' is provided (in `xc/programs/Xserver/hw/xfree86/etc/` in the source tree, and in `/usr/X11R6/lib/X11/etc/` in the binary kit). You can compile and install this program. Then to make use of it, add the line `` `VTInit "2key off"' `` to the Keyboard section of your `XF86Config` file to cause the program to be run automatically when the server starts up. Doing this means that 2-key switching will be turned off while in the server's VT, but will still be on for the other VTs.

For further details, refer to the keyboard(7) man page included with the release notes (the on-line man page doesn't have this information).

3. Running SVR3 binaries on SVR4.0.4 and SVR4.2

SVR4.0.4 added the `Advanced Compatibility Package', which provides iBCS-2 compliance for running SVR3 binaries. These facilities are also present in SVR4.2. XFree86 makes use of this to accept local connections from SVR3 clients. The XFree86 binary distribution is built to use these capabilities. You need to install the `Advanced Compatibility Package', if you have not done so already.

We have found that SVR4.0.4 is not able to run all SCO, and perhaps not many ISC SVR3 binaries. This is not a failing of XFree86, but of SVR4 itself. One particular example is that many SVR3 programs are ignorant of the UFS filesystem, and attempt to read directories as files, rather than using the system call that is defined for the purpose. This will fail for obvious reasons. The SVR4.0.4 release notes from USL (which you should have gotten from your vendor) have lots of suggestions for how to improve compatibility.

That said, we have had luck with several SCO binaries right out of the box. No changes are needed - just go to an xterm window and run the program.

ISC users will need a binary editor before they can attempt to run their binaries. ISC, for whatever reason, put the pipe for local connections in `/tmp/.X11-unix/Xn`. This unfortunately is the same place as the X Consortium X server puts the Unix-domain socket normally used for local connections. The XFree86 server was modified to use `/dev/X/ISCCONN/Xn` for local connections to ISC clients. So what you must do is use a binary editor to edit your client program. Search for `/tmp/.X11-unix`, and change it to `/dev/X/ISCCONN`. Now you just have to worry about base-OS compatibility.

4. Notes for building XFree86 on SVR4

1. If you are using gcc with SVR4, we highly recommend that you use gcc-2.4.5 (or a later stable release). Version 2.6.0 has some problems on i386 platforms and is not recommended.
2. It is recommended that you increase the `UFSNINODE` (for a UFS filesystem) and/or the `S5NINODE` (for an S5 filesystem) kernel parameter to about 650 before attempting to build the distribution. See the "Notes for running XFree86 on SVR4" section for some other parameters you may want to change.
3. The `BOOTSTRAPCFLAGS` required are:

   For Unixware: "`-DUSL`" For NCR: "`-DNCR`" For other SVR4: "`-DSVR4 -Di386`"

5. Notes for running XFree86 on SVR4

NOTE: If you intend to use any of the accelerated servers, read section 10 and follow the instructions. Otherwise the X server will crash when exiting, restarting, or switching VTs.

1. For SVR4, you may also need to add `/usr/X11R6/lib` to your `LD_LIBRARY_PATH`, but this is not required for running properly built clients.
2. You may want to increase some kernel parameters (either by running `idtune`, or editing `/etc/conf/cf.d/stune`, and rebuilding the kernel with `idbuild`):

   `[HS]FNOLIM` hard/soft limit for number of open files

   `MAXUP` max number of processes per user

   `ARG_MAX` max length of an arg list

   `SHMMAX` max size of shared memory segment(in bytes)
3. Choose which mouse driver you will use. For SVR4 the best choice depends on which version you are using. If you have a bus mouse then Xqueue is probably the only option. For a serial mouse the options are as follows:

   Esix 4.0.3

   Xqueue works. It is also possible to use the standard asy driver directly, but the mouse operation is "jerky".

   Microport SVR4 [34].1

   Xqueue works fine, and the asy driver can also be used directly giving smooth mouse operation.

   To use Xqueue, set the `Protocol` to `Xqueue` in both the `Keyboard` and `Pointer` sections of your `XF86Config` file, and You must have the mouse driver package installed, and must run mouseadmin to set it up for your mouse. If mouseadmin won't work try doing `` `touch /dev/gmse' `` before running it. (Note that mouseadmin will need to be rerun after rebuilding a kernel unless you add an appropriate entry to `/etc/conf/node.d/gmse`.)

   NOTE: Many of the accelerated server/drivers have problems when using a HW cursor and Xqueue together. If you have a serial mouse, you can work around this by not using Xqueue. Otherwise the only workaround is to disable the HW cursor. This is done by adding the line:

   ```
   Option "sw_cursor"
   ```

   to the Device section of your XF86Config file. The S3 server is the only one known to not have this problem.

If you have problems with both Xqueue and your standard asy driver with SVR4, then you sh install SAS. When using SAS, set up `XF86Config` as you would for the standard driver.

SAS is available from ftp.physics.su.oz.au. When using SAS for a serial mouse, you will get smoother operation if you change `EVENT_TIME` from 80 to 30 in `sas.h`. A couple of details which aren't spelled out in the SAS README are:

- An example of the line you should add to `/etc/ap/chan.ap` is:

```
MAJOR    0       255     ldterm ttcompat
```

where `MAJOR` is replaced by the major number used for SAS devices. To determine what that is, check `/etc/conf/cf.d/mdevice` after rebuilding the kernel. The major number is the sixth field in the line starting with `sas'. This file must be updated before rebooting with the new kernel.

- The installation instructions omit the following:

    3a) Disable the asy driver by either running `` `kconfig' `` or editing `/etc/conf/sdevice.d/asy`.

    3b) Rebuild the kernel by running `/etc/conf/bin/idbuild`

4. If you want to use xdm with SVR4, extract the files from the shar file in `/usr/X11R6/lib/X11/etc/XdmConf.svr4` into a temporary directory. The `README` file tells where the individual files should be installed. Be sure to read through each file and make any site-specific changes that you need.

   NOTE: Some SVR4 versions (one example is Esix 4.0.3) have a default inittab which runs `vtgetty' on the console. This does not work well when starting xdm at boot time. The problem is that when you logout from a vtgetty session it wants to close all the VTs -- including the one xdm is using for the server. It is recommended that you use `getty'. If you change `/etc/inittab`, remember to also change `/etc/conf/cf.d/init.base` or you will lose the changes when you next rebuild the kernel.
5. If you want to change the number of VTs available on SVR4, just edit the file `/etc/default/workstations` and change the number there. The device nodes will be created/deleted next time you reboot.
6. The default local connection types have changed in X11R6. Unix domain sockets are no longer treated as a "local" connection type. This means that a client connecting to :0 will use not use a Unix socket for the connection. To use the Unix socket connection, the client must connect to unix:0.

   The local connection types available are "`NAMED`" (named streams pipe), "`PTS`" (old-stype USL streams pipe), "`SCO`" (SCO Xsight streams pipe), and "`ISC`" (ISC streams pipe). The `XLOCAL` environment variable can be used to set which types of local connection should be used in order of preference. The default setting is `PTS:NAMED:ISC:SCO`. It is recommended that `NAMED` be used in most cases because it is faster than the default `PTS`, and because using PTS can cause you to run out of `/dev/pts/` devices (each client using PTS requires a `/dev/pts` device). To set up the default local connection type, make sure that `XLOCAL` is set and exported in your `.xinitrc` file (when using xinit or startx) or your `/usr/X11R6/lib/xdm/Xsession` script

(when using xdm).

TOC

Information for SVR4 Users **:** *Notes for running XFree86 on SVR4*
Previous: *Notes for building XFree86 on SVR4*
Next: *Building non-core clients with SVR4*

6. Building non-core clients with SVR4

1. A lot of clients (even some which have explicit SVR4 support) require `-DSYSV` when building under SVR4. This will not be set when using the default configuration. A quick fix is to add something like the following to the client's Imakefile:

   ```
   #if SystemV4
        DEFINES = -DSYSV OTHER_CLIENT_DEPENDENT_DEFINES
   #endif
   ```

   The best solution is to modify the code so it compiles correctly without `-DSYSV`.

7. Using DOS/Merge 2.2 with XFree86

It is possible to use the Locus DOS/Merge 2.2 X clients with XFree86. You need to do a couple of things for this to work, though. One change is a generic problem with the X client and X11R5/6; the others are to work with some things that are specific to the XFree86 servers. Here are the things you need to do:

1. Set and export `$XMERGE` in your `.xinitrc` and/or `.xsession` files. In general, you should set `XMERGE=vga`.
2. You MUST use the "xqueue" driver instead of the server's native keyboard and mouse driver, if you intend to use the "zoom" feature of the `dos' client. Otherwise the mouse will cease to function after the first time you "zoom" (because the `dos' client uses the native driver, and the server will not be able to access the mouse after the zoom ends). The only other alternative is to use separate mice on separate devices.
3. You need to install the `dos' client fonts in the XFree86 font directories. Locate the BDF files (search for files with names matching the pattern `*pc???.bdf'). These will likely be `/usr/lib/X11/fonts/misc`. Go to the directory where these files are located, and execute the following (using `sh' or `ksh'):

```
for i in *pc???.bdf
do
    /usr/X11R6/bin/bdftopcf $i > \
                /usr/X11R6/lib/X11/fonts/misc/`basename $i .bdf`.pcf
done
cd /usr/X11R6/lib/X11/fonts/misc
/usr/X11R6/bin/mkfontdir
# Do this only if the server is already running.
/usr/X11R6/bin/xset fp rehash
```

4. The `dos' client program uses a translation table to map from an internal key representation to the X keymap. It is likely that the table supplied with Merge 2.2 use the mapping for SCO's server. A correct mapping table is available in `/usr/X11R6/lib/X11/etc/xcode.xfree86`. This file should be installed in `/usr/lib/merge/xc`. In addition, you must add the following resource to the `dos' client's application-defaults file (usually in `/usr/lib/X11/app-defaults/DOS`):

```
dos*xcodetable: /usr/lib/merge/xc/xcode.xfree86
```

   It will be obvious if this new code table is needed, as the arrow keys on the keypad will fail to function in the `dos' client if the wrong table is installed.
5. For the "zoom" feature to work correctly, you must run `dos' with $DISPLAY set to "unix:N" or "host_name:N". If you use just ":0", the client will not function properly. `dos' does not accept a `-display' parameter. Hence it is probably a good idea to replace the `dos' program with something like this:

```
#!/usr/bin/ksh
if [ "X${DISPLAY}" != "X" ]
then
        case ${DISPLAY} in
        :*)
                DISPLAY=unix${DISPLAY}
                ;;
```

```
        esac
fi
/usr/bin/dos.real "$@"
```

TOC

Information for SVR4 Users : *Using DOS/Merge 2.2 with XFree86*
Previous: *Building non-core clients with SVR4*
Next: *Keyboard mapping problems with some Esix systems*

8. Keyboard mapping problems with some Esix systems

One of the console driver patches for Esix 4.0.3A causes the XFree86 server's default keymap to be corrupted. If you are being affected by this problem it will be obvious because few (if any) of the keys will be mapped correctly. There are two solutions to this. One is to remove the console driver patch which introduced the problem. The second is to use xmodmap(1) to reset the default mapping after server startup. The default mapping is provided in the file `/usr/X11R6/lib/X11/etc/xmodmap.std`, and can be installed automatically by adding the line:

```
xmodmap /usr/X11R6/lib/X11/etc/xmodmap.std
```

to your `.xinitrc` file (or your `Xsetup` file if using xdm).

9. 106 Japanese keyboard problem on PANIX

PANIX for PC-AT uses Japanese keycodes standardized by DICOP(Desktop UNIX for Intel Cooperative Promotion Group) in Japan. Therefore keycode confliction occurs with 106 Japanese keyboard in XFree86. To avoid it, specify the keyword "panix106" in XF86Config like this:

```
Section "Keyboard"
  Protocol    "Standard"
  Autorepeat  500 5
  XkbModel    "jp106"
  XkbLayout   "jp"
  panix106
EndSection
```

10. A kernel patch that is required for accelerated servers

SVR4.0 has a bug handling programs that access extended I/O registers (above 0x3FF). Boards like S3 and 8514/A use these extended I/O registers. XFree86 supports boards that tickle this bug. In preparation for using these servers, we have produced a kernel patch that works around the problem, and provided scripts for you that will both install and back out the patch. You must install this if you intend to use the S3, 8514, Mach8, Mach32, P9000, AGX or W32 servers.

Dell 2.2 is known to not need the patch, because Thomas Roell found and fixed the bug while he was working for Dell. Microport has fixed this in their 4.0 v4.2 release. Also, SVR4.2 does not need this patch, as the problem has been fixed by USL.

The patch scripts are located in `xc/programs/Xserver/hw/xfree86/etc` in the source tree, and `/usr/X11R6/lib/X11/etc` in the binary distribution. The files are `svr4_patch' to install the patch, and `svr4_patch_rem' to back it out. The file that is being patched is `/etc/conf/pack.d/kernel/os.o`. The patch script verifies the presence of the bug before patching, and will tell you whether or not it succeeded in patching. You need to run the `svr4_patch' script as root, obviously. The original `os.o` file, as well as the patching program, and a copy of the removal script are stored in the directory `/etc/conf/pack.d/kernel/.xfree86`

Thanks to John M. Sully of Microport for helping us find a simple workaround for this problem, and giving us permission to release the information.

11. Other problems

Some accelerated drivers may cause the machine to lockup when starting up the server on some versions of SVR4.0. The problem seems to be related to the kernel checking for the presence of physical memory when mmaping /dev/pmem. This can cause problems when mapping memory mapped registers. This was known to be a problem with the MGA driver in the SVGA server. Some other drivers may be affected too. The problem with the MGA driver is now fixed.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/SVR4.sgml,v 3.13.2.3 1998/02/28
08:54:11 dawes Exp $
```

```
$XConsortium: SVR4.sgml /main/8 1996/10/27 11:06:06 kaleb $
```

XFree86 Video Timings HOWTO

TOC

Eric S. Raymond <esr@thyrsus.com>

Version 3.0, 8 Aug 1997

How to compose a mode line for your card/monitor combination under XFree86. The XFree86 distribution now includes good facilities for configuring most standard combinations; this document is mainly useful if you are tuning a custom mode line for a high-performance monitor or very unusual hardware. It may also help you in using xvidtune to tweak a standard mode that is not quite right for your monitor.

---

1. Disclaimer
2. Introduction
3. How Video Displays Work
4. Basic Things to Know about your Display and Adapter
   - 4.1. The monitor's video bandwidth:
   - 4.2. What these control:
5. Interpreting the Basic Specifications
   - 5.1. About Bandwidth:
   - 5.2. Sync Frequencies and the Refresh Rate:
6. Tradeoffs in Configuring your System
7. Memory Requirements
8. Computing Frame Sizes
9. Black Magic and Sync Pulses
   - 9.1. Horizontal Sync:
   - 9.2. Vertical Sync:

10. Putting it All Together

11. Overdriving Your Monitor

12. Using Interlaced Modes

13. Questions and Answers

14. Fixing Problems with the Image.

14.1. The image is displaced to the left or right

14.2. The image is displaced up or down

14.3. The image is too large both horizontally and vertically

14.4. The image is too wide (too narrow) horizontally

14.5. The image is too deep (too shallow) vertically

15. Plotting Monitor Capabilities

16. Credits

1. Disclaimer

You use the material herein SOLELY AT YOUR OWN RISK. It is possible to harm both your monitor and yourself when driving it outside the manufacturer's specs. Read Overdriving Your Monitor for detailed cautions. Any damages to you or your monitor caused by overdriving it are your problem.

The most up-to-date version of this HOWTO can be found at the Linux Documentation Project web page.

Please direct comments, criticism, and suggestions for improvement to esr@snark.thyrsus.com. Please do *not* send email pleading for a magic solution to your special monitor problem, as doing so will only burn up my time and frustrate you -- everything I know about the subject is already in here.

2. Introduction

The XFree86 server allows users to configure their video subsystem and thus encourages best use of existing hardware. This tutorial is intended to help you learn how to generate your own timing numbers to make optimum use of your video card and monitor.

We'll present a method for getting something that works, and then show you how you can experiment starting from that base to develop settings that optimize for your taste.

Starting with XFree86 3.2, XFree86 provides an **XF86Setup**(1) program that makes it easy to generate a working monitor mode interactively, without messing with video timing number directly. So you shouldn't actually need to calculate a base monitor mode in most cases. Unfortunately, **XF86Setup**(1) has some limitations; it only knows about standard video modes up to 1280x1024. If you have a very high-performance monitor capable of 1600x1200 or more you will still have to compute your base monitor mode yourself.

Recent versions of XFree86 provide a tool called **xvidtune**(1) which you will probably find quite useful for testing and tuning monitor modes. It begins with a gruesome warning about the possible consequences of mistakes with it. If you pay careful attention to this document and learn what is behind the pretty numbers in xvidtune's boxes, you will become able to use xvidtune effectively and with confidence.

If you already have a mode that almost works (in particular, if one of predefined VESA modes gives you a stable display but one that's displaced right or left, or too small, or too large) you can go straight to the section on Fixing Problems with the Image. This will enlighten you on ways to tweak the timing numbers to achieve particular effects.

If you have **xvidtune**(1), you'll be able to test new modes on the fly, without modifying your X configuration files or even rebooting your X server. Otherwise, XFree86 allows you to hot-key between different modes defined in Xconfig (see XFree86.man for details). Use this capability to save yourself hassles! When you want to test a new mode, give it a unique mode label and add it to the *end* of your hot-key list. Leave a known-good mode as the default to fall back on if the test mode doesn't work.

3. How Video Displays Work

Knowing how the display works is essential to understanding what numbers to put in the various fields in the file Xconfig. Those values are used in the lowest levels of controlling the display by the XFree86 server.

The display generates a picture from a series of dots. The dots are arranged from left to right to form lines. The lines are arranged from top to bottom to form the picture. The dots emit light when they are struck by the electron beam inside the display. To make the beam strike each dot for an equal amount of time, the beam is swept across the display in a constant pattern.

The pattern starts at the top left of the screen, goes across the screen to the right in a straight line, and stops temporarily on the right side of the screen. Then the beam is swept back to the left side of the display, but down one line. The new line is swept from left to right just as the first line was. This pattern is repeated until the bottom line on the display has been swept. Then the beam is moved from the bottom right corner of the display to the top left corner, and the pattern is started over again.

There is one variation of this scheme known as interlacing: here only every second line is swept during one half-frame and the others are filled in in during a second half-frame.

Starting the beam at the top left of the display is called the beginning of a frame. The frame ends when the beam reaches the the top left corner again as it comes from the bottom right corner of the display. A frame is made up of all of the lines the beam traced from the top of the display to the bottom.

If the electron beam were on all of the time it was sweeping through the frame, all of the dots on the display would be illuminated. There would be no black border around the edges of the display. At the edges of the display the picture would become distorted because the beam is hard to control there. To reduce the distortion, the dots around the edges of the display are not illuminated by the beam even though the beam may be pointing at them. The viewable area of the display is reduced this way.

Another important thing to understand is what becomes of the beam when no spot is being painted on the visible area. The time the beam would have been illuminating the side borders of the display is used for sweeping the beam back from the right edge to the left and moving the beam down to the next line. The time the beam would have been illuminating the top and bottom borders of the display is used for moving the beam from the bottom-right corner of the display to the top-left corner.

The adapter card generates the signals which cause the display to turn on the electron beam at each dot to generate a picture. The card also controls when the display moves the beam from the right side to the left and down a line by generating a signal called the horizontal sync (for synchronization) pulse. One horizontal sync pulse occurs at the end of every line. The adapter also generates a vertical sync pulse which signals the display to move the beam to the top-left corner of the display. A vertical sync pulse is generated near the end of every frame.

The display requires that there be short time periods both before and after the horizontal and vertical pulses so that the position of the electron beam can stabilize. If the beam can't stabilize, the picture will not be steady.

In a later section, we'll come back to these basics with definitions, formulas and examples to help you use them.

---

4. Basic Things to Know about your Display and Adapter

There are some fundamental things you need to know before hacking an Xconfig entry. These are:

- your monitor's horizontal and vertical sync frequency options
- your video adapter's driving clock frequency, or "dot clock"
- your monitor's bandwidth

The monitor sync frequencies:

The horizontal sync frequency is just the number of times per second the monitor can write a horizontal scan line; it is the single most important statistic about your monitor. The vertical sync frequency is the number of times per second the monitor can traverse its beam vertically.

Sync frequencies are usually listed on the specifications page of your monitor manual. The vertical sync frequency number is typically calibrated in Hz (cycles per second), the horizontal one in KHz (kilocycles per second). The usual ranges are between 50 and 150Hz vertical, and between 31 and 135KHz horizontal.

If you have a multisync monitor, these frequencies will be given as ranges. Some monitors, especially lower-end ones, have multiple fixed frequencies. These can be configured too, but your options will be severely limited by the built-in monitor characteristics. Choose the highest frequency pair for best resolution. And be careful --- trying to clock a fixed-frequency monitor at a higher speed than it's designed for can easily damage it.

Earlier versions of this guide were pretty cavalier about overdriving multisync monitors, pushing them past their nominal highest vertical sync frequency in order to get better performance. We have since had more reasons pointed out to us for caution on this score; we'll cover those under Overdriving Your Monitor below.

The card driving clock frequency:

Your video adapter manual's spec page will usually give you the card's dot clock (that is, the total number of pixels per second it can write to the screen). If you don't have this information, the X server will get it for you. Even if your X locks up your monitor, it will emit a line of clock and other info to standard output. If you redirect this to a file, it should be saved even if you have to reboot to get your console back. (Recent versions of the X servers all support a --probeonly option that prints out this information and exits without actually starting up X or changing the video mode.)

Your X startup message should look something like one of the following examples:

If you're using XFree86:

```
Xconfig: /usr/X11R6/lib/X11/Xconfig
(**) stands for supplied, (--) stands for probed/default values
(**) Mouse: type: MouseMan, device: /dev/ttyS1, baudrate: 9600
Warning: The directory "/usr/andrew/X11fonts" does not exist.
         Entry deleted from font path.
(**) FontPath set to "/usr/lib/X11/fonts/misc/,/usr/lib/X11/fonts/75dpi/"
(--) S3: card type: 386/486 localbus
(--) S3: chipset:   924
                    ---
    Chipset -- this is the exact chip type; an early mask of the 86C911

(--) S3: chipset driver: s3_generic
```


```
(--) S3: videoram:  1024k
                    -----
         Size of on-board frame-buffer RAM

(**) S3: clocks:  25.00  28.00  40.00   3.00  50.00  77.00  36.00  45.00
(**) S3: clocks:   0.00   0.00  79.00  31.00  94.00  65.00  75.00  71.00
                  ------------------------------------------------------
                              Possible driving frequencies in MHz

(--) S3: Maximum allowed dot-clock: 110MHz
                                    ------
                                  Bandwidth
(**) S3: Mode "1024x768": mode clock =  79.000, clock used =  79.000
(--) S3: Virtual resolution set to 1024x768
(--) S3: Using a banksize of 64k, line width of 1024
(--) S3: Pixmap cache:
(--) S3: Using 2 128-pixel 4 64-pixel and 8 32-pixel slots
(--) S3: Using 8 pages of 768x255 for font caching
```

If you're using SGCS or X/Inside X:

```
WGA: 86C911 (mem: 1024k clocks: 25 28 40 3 50 77 36 45 0 0 79 31 94 65 75 71)
---  ------       -----         ----------------------------------------
 |     |            |                 Possible driving frequencies in MHz
 |     |            +-- Size of on-board frame-buffer RAM
 |     +-- Chip type
 +-- Server type
```

Note: do this with your machine unloaded (if at all possible). Because X is an application, its timing loops can collide with disk activity, rendering the numbers above inaccurate. Do it several times and watch for the numbers to stabilize; if they don't, start killing processes until they do. SVr4 users: the mousemgr process is particularly likely to mess you up.

In order to avoid the clock-probe inaccuracy, you should clip out the clock timings and put them in your Xconfig as the value of the Clocks property --- this suppresses the timing loop and gives X an exact list of the clock values it can try. Using the data from the example above:

```
wga
        Clocks  25 28 40 3 50 77 36 45 0 0 79 31 94 65 75 71
```

On systems with a highly variable load, this may help you avoid mysterious X startup failures. It's possible for X to come up, get its timings wrong due to system load, and then not be able to find a matching dot clock in its config database --- or find the wrong one!

4.1. The monitor's video bandwidth:

If you're running XFree86, your server will probe your card and tell you what your highest-available dot clock is.

Otherwise, your highest available dot clock is approximately the monitor's video bandwidth. There's a lot of give here, though --- some monitors can run as much as 30% over their nominal bandwidth. The risks here have to do with exceeding the monitor's rated vertical-sync frequency; we'll discuss them in detail below.

Knowing the bandwidth will enable you to make more intelligent choices between possible configurations. It may affect your display's visual quality (especially sharpness for fine details).

Your monitor's video bandwidth should be included on the manual's spec page. If it's not, look at the monitor's highest

rated resolution. As a rule of thumb, here's how to translate these into bandwidth estimates (and thus into rough upper bounds for the dot clock you can use):

640x480	25
800x600	36
1024x768	65
1024x768 interlaced	45
1280x1024	110
1600x1200	185

BTW, there's nothing magic about this table; these numbers are just the lowest dot clocks per resolution in the standard XFree86 Modes database (except for the last, which I interpolated). The bandwidth of your monitor may actually be higher than the minimum needed for its top resolution, so don't be afraid to try a dot clock a few MHz higher.

Also note that bandwidth is seldom an issue for dot clocks under 65MHz or so. With an SVGA card and most hi-res monitors, you can't get anywhere near the limit of your monitor's video bandwidth. The following are examples:

Brand	Video Bandwidth
NEC 4D	75Mhz
Nano 907a	50Mhz
Nano 9080i	60Mhz
Mitsubishi HL6615	110Mhz
Mitsubishi Diamond Scan	100Mhz
IDEK MF-5117	65Mhz
IOCOMM Thinksync-17 CM-7126	136Mhz
HP D1188A	100Mhz
Philips SC-17AS	110Mhz
Swan SW617	85Mhz
Viewsonic 21PS	185Mhz

Even low-end monitors usually aren't terribly bandwidth-constrained for their rated resolutions. The NEC Multisync II makes a good example --- it can't even display 800x600 per its spec. It can only display 800x560. For such low resolutions you don't need high dot clocks or a lot of bandwidth; probably the best you can do is 32Mhz or 36Mhz, both of them are still not too far from the monitor's rated video bandwidth of 30Mhz.

At these two driving frequencies, your screen image may not be as sharp as it should be, but definitely of tolerable quality. Of course it would be nicer if NEC Multisync II had a video bandwidth higher than, say, 36Mhz. But this is not critical for common tasks like text editing, as long as the difference is not so significant as to cause severe image distortion (your eyes would tell you right away if this were so).

4.2. What these control:

The sync frequency ranges of your monitor, together with your video adapter's dot clock, determine the ultimate resolution that you can use. But it's up to the driver to tap the potential of your hardware. A superior hardware combination without an equally competent device driver is a waste of money. On the other hand, with a versatile device driver but less capable hardware, you can push the hardware's envelope a little. This is the design philosophy of XFree86.

5. Interpreting the Basic Specifications

This section explains what the specifications above mean, and some other things you'll need to know. First, some definitions. Next to each in parens is the variable name we'll use for it when doing calculations

horizontal sync frequency (HSF)

Horizontal scans per second (see above).

vertical sync frequency (VSF)

Vertical scans per second (see above). Mainly important as the upper limit on your refresh rate.

dot clock (DCF)

More formally, `driving clock frequency'; The frequency of the crystal or VCO on your adaptor --- the maximum dots-per-second it can emit.

video bandwidth (VB)

The highest frequency you can feed into your monitor's video input and still expect to see anything discernible. If your adaptor produces an alternating on/off pattern, its lowest frequency is half the DCF, so in theory bandwidth starts making sense at DCF/2. For tolerably crisp display of fine details in the video image, however, you don't want it much below your highest DCF, and preferably higher.

frame length (HFL, VFL)

Horizontal frame length (HFL) is the number of dot-clock ticks needed for your monitor's electron gun to scan one horizontal line, *including the inactive left and right borders*. Vertical frame length (VFL) is the number of scan lines in the *entire* image, including the inactive top and bottom borders.

screen refresh rate (RR)

The number of times per second your screen is repainted (this is also called "frame rate"). Higher frequencies are better, as they reduce flicker. 60Hz is good, VESA-standard 72Hz is better. Compute it as

```
RR = DCF / (HFL * VFL)
```

Note that the product in the denominator is *not* the same as the monitor's visible resolution, but typically somewhat larger. We'll get to the details of this below.

The rates for which interlaced modes are usually specified (like 87Hz interlaced) are actually the half-frame rates: an entire screen seems to have about that flicker frequency for typical displays,

but every single line is refreshed only half as often.

For calculation purposes we reckon an interlaced display at its full-frame (refresh) rate, i.e. 43.5Hz. The quality of an interlaced mode is better than that of a non-interlaced mode with the same full-frame rate, but definitely worse then the non-interlaced one corresponding to the half-frame rate.

5.1. About Bandwidth:

Monitor makers like to advertise high bandwidth because it constrains the sharpness of intensity and color changes on the screen. A high bandwidth means smaller visible details.

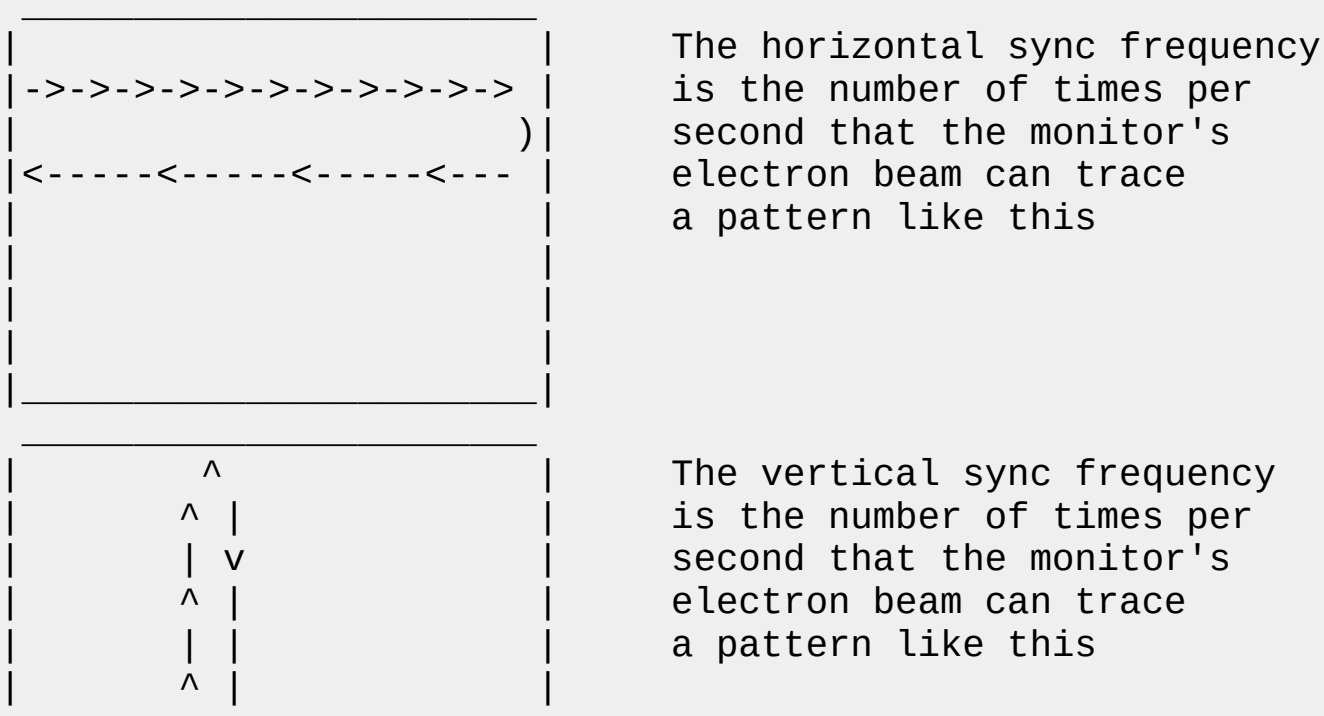
Your monitor uses electronic signals to present an image to your eyes. Such signals always come in in wave form once they are converted into analog form from digitized form. They can be considered as combinations of many simpler wave forms each one of which has a fixed frequency, many of them are in the Mhz range, eg, 20Mhz, 40Mhz, or even 70Mhz. Your monitor video bandwidth is, effectively, the highest-frequency analog signal it can handle without distortion.

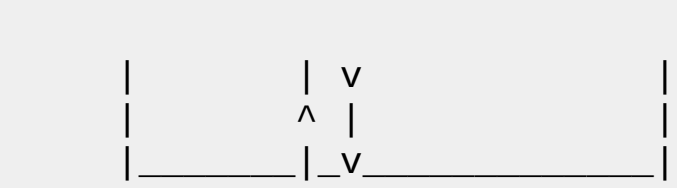
For our purposes, bandwidth is mainly important as an approximate cutoff point for the highest dot clock you can use.

5.2. Sync Frequencies and the Refresh Rate:

Each horizontal scan line on the display is just the visible portion of a frame-length scan. At any instant there is actually only one dot active on the screen, but with a fast enough refresh rate your eye's persistence of vision enables you to "see" the whole image.

Here are some pictures to help:



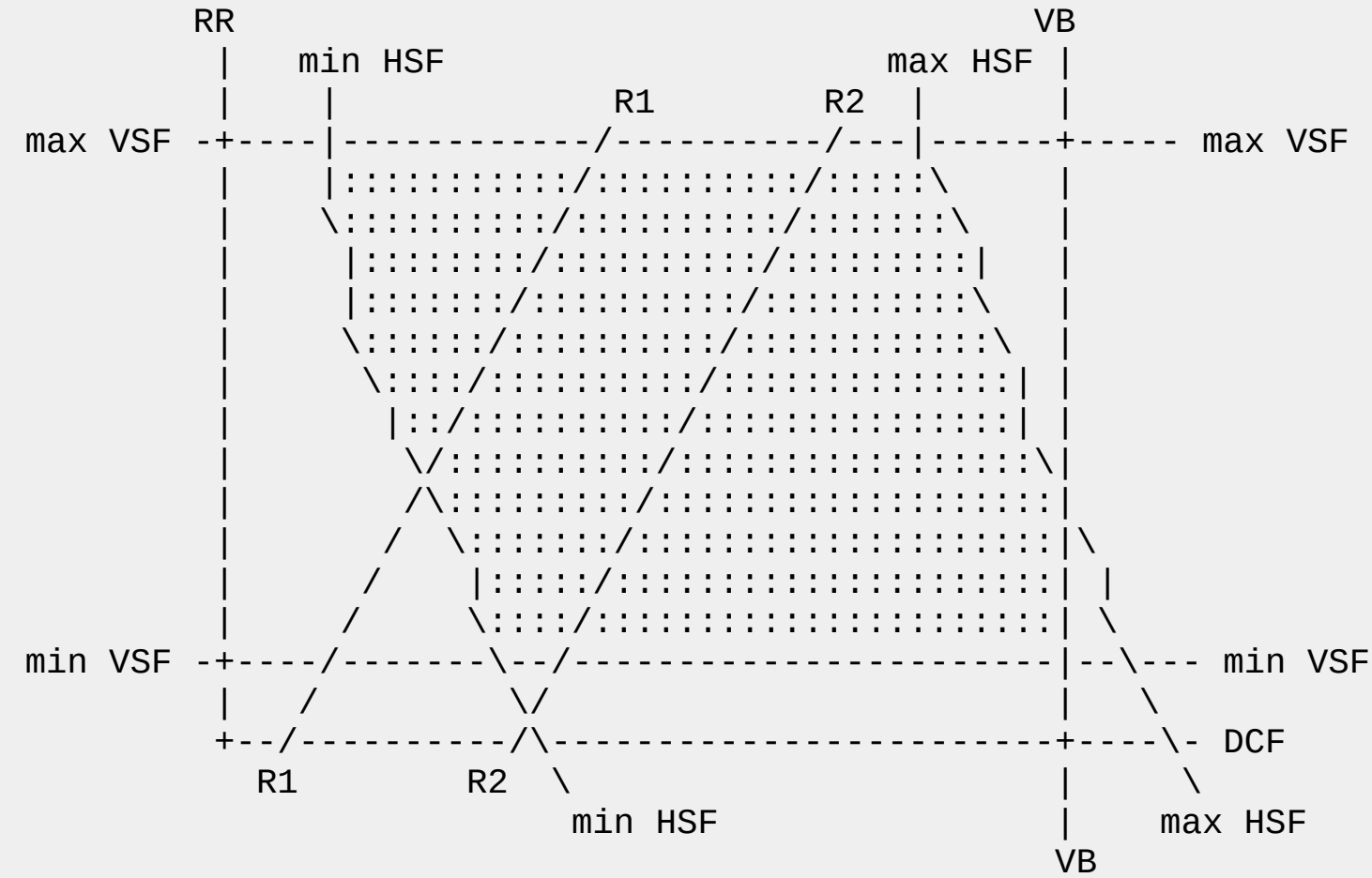


Remember that the actual raster scan is a very tight zigzag pattern; that is, the beam moves left-right and at the same time up-down.

Now we can see how the dot clock and frame size relates to refresh rate. By definition, one hertz (hz) is one cycle per second. So, if your horizontal frame length is HFL and your vertical frame length is VFL, then to cover the entire screen takes (HFL * VFL) ticks. Since your card emits DCF ticks per second by definition, then obviously your monitor's electron gun(s) can sweep the screen from left to right and back and from bottom to top and back DCF / (HFL * VFL) times/sec. This is your screen's refresh rate, because it's how many times your screen can be updated (thus *refreshed*) per second!

You need to understand this concept to design a configuration which trades off resolution against flicker in whatever way suits your needs.

For those of you who handle visuals better than text, here is one:



This is a generic monitor mode diagram. The x axis of the diagram shows the clock rate (DCF), the y axis represents the refresh rate (RR). The filled region of the diagram describes the monitor's capabilities:

every point within this region is a possible video mode.

The lines labeled `R1' and `R2' represent a fixed resolutions (such as 640x480); they are meant to illustrate how one resolution can be realized by many different combinations of dot clock and refresh rate. The R2 line would represent a higher resolution than R1.

The top and bottom boundaries of the permitted region are simply horizontal lines representing the limiting values for the vertical sync frequency. The video bandwidth is an upper limit to the clock rate and hence is represented by a vertical line bounding the capability region on the right.

Under Plotting Monitor Capabilities) you'll find a program that will help you plot a diagram like this (but much nicer, with X graphics) for your individual monitor. That section also discusses the interesting part; the derivation of the boundaries resulting from the limits on the horizontal sync frequency.

6. Tradeoffs in Configuring your System

Another way to look at the formula we derived above is

```
DCF = RR * HFL * VFL
```

That is, your dot clock is fixed. You can use those dots per second to buy either refresh rate, horizontal resolution, or vertical resolution. If one of those increases, one or both of the others must decrease.

Note, though, that your refresh rate cannot be greater than the maximum vertical sync frequency of your monitor. Thus, for any given monitor at a given dot clock, there is a minimum product of frame lengths below which you can't force it.

In choosing your settings, remember: if you set RR too low, you will get mugged by screen flicker.

You probably do not want to pull your refresh rate below 60Hz. This is the flicker rate of fluorescent lights; if you're sensitive to those, you need to hang with 72Hz, the VESA ergonomic standard.

Flicker is very eye-fatiguing, though human eyes are adaptable and peoples' tolerance for it varies widely. If you face your monitor at a 90% viewing angle, are using a dark background and a good contrasting color for foreground, and stick with low to medium intensity, you *may* be comfortable at as little as 45Hz.

The acid test is this: open a xterm with pure white back-ground and black foreground using `xterm -bg white -fg black` and make it so large as to cover the entire viewable area. Now turn your monitor's intensity to 3/4 of its maximum setting, and turn your face away from the monitor. Try peeking at your monitor sideways (bringing the more sensitive peripheral-vision cells into play). If you don't sense any flicker or if you feel the flickering is tolerable, then that refresh rate is fine with you. Otherwise you better configure a higher refresh rate, because that semi-invisible flicker is going to fatigue your eyes like crazy and give you headaches, even if the screen looks OK to normal vision.

For interlaced modes, the amount of flicker depends on more factors such as the current vertical resolution and the actual screen contents. So just experiment. You won't want to go much below about 85Hz half frame rate, though.

So let's say you've picked a minimum acceptable refresh rate. In choosing your HFL and VFL, you'll have some room for maneuver.

7. Memory Requirements

Available frame-buffer RAM may limit the resolution you can achieve on color or gray-scale displays. It probably isn't a factor on displays that have only two colors, white and black with no shades of gray in between.

For 256-color displays, a byte of video memory is required for each visible dot to be shown. This byte contains the information that determines what mix of red, green, and blue is generated for its dot. To get the amount of memory required, multiply the number of visible dots per line by the number of visible lines. For a display with a resolution of 800x600, this would be 800 x 600 = 480,000, which is the number of visible dots on the display. This is also, at one byte per dot, the number of bytes of video memory that are necessary on your adapter card.

Thus, your memory requirement will typically be (HR * VR)/1024 Kbytes of VRAM, rounded up. If you have more memory than strictly required, you'll have extra for virtual-screen panning.

However, if you only have 512K on board, then you can't use this resolution. Even if you have a good monitor, without enough video RAM, you can't take advantage of your monitor's potential. On the other hand, if your SVGA has one meg, but your monitor can display at most 800x600, then high resolution is beyond your reach anyway (see Using Interlaced Modes for a possible remedy).

Don't worry if you have more memory than required; XFree86 will make use of it by allowing you to scroll your viewable area (see the Xconfig file documentation on the virtual screen size parameter). Remember also that a card with 512K bytes of memory really doesn't have 512,000 bytes installed, it has 512 x 1024 = 524,288 bytes.

If you're running SGCS X (now called X/Inside) using an S3 card, and are willing to live with 16 colors (4 bits per pixel), you can set depth 4 in Xconfig and effectively double the resolution your card can handle. S3 cards, for example, normally do 1024x768x256. You can make them do 1280x1024x16 with depth 4.

8. Computing Frame Sizes

Warning: this method was developed for multisync monitors. It will probably work with fixed-frequency monitors as well, but no guarantees!

Start by dividing DCF by your highest available HSF to get a horizontal frame length.

For example; suppose you have a Sigma Legend SVGA with a 65MHz dot clock, and your monitor has a 55KHz horizontal scan frequency. The quantity (DCF / HSF) is then 1181 (65MHz = 65000KHz; 65000/55 = 1181).

Now for our first bit of black magic. You need to round this figure to the nearest multiple of 8. This has to do with the VGA hardware controller used by SVGA and S3 cards; it uses an 8-bit register, left-shifted 3 bits, for what's really an 11-bit quantity. Other card types such as ATI 8514/A may not have this requirement, but we don't know and the correction can't hurt. So round the usable horizontal frame length figure down to 1176.

This figure (DCF / HSF rounded to a multiple of 8) is the minimum HFL you can use. You can get longer HFLs (and thus, possibly, more horizontal dots on the screen) by setting the sync pulse to produce a lower HSF. But you'll pay with a slower and more visible flicker rate.

As a rule of thumb, 80% of the horizontal frame length is available for horizontal resolution, the visible part of the horizontal scan line (this allows, roughly, for borders and sweepback time -- that is, the time required for the beam to move from the right screen edge to the left edge of the next raster line). In this example, that's 944 ticks.

Now, to get the normal 4:3 screen aspect ratio, set your vertical resolution to 3/4ths of the horizontal resolution you just calculated. For this example, that's 708 ticks. To get your actual VFL, multiply that by 1.05 to get 743 ticks.

The 4:3 is not technically magic; nothing prevents you from using a non-Golden-Section ratio if that will get the best use out of your screen real estate. It does make figuring frame height and frame width from the diagonal size convenient, you just multiply the diagonal by by 0.8 to get width and 0.6 to get height.

So, HFL=1176 and VFL=743. Dividing 65MHz by the product of the two gives us a nice, healthy 74.4Hz refresh rate. Excellent! Better than VESA standard! And you got 944x708 to boot, more than the 800 by 600 you were probably expecting. Not bad at all!

You can even improve the refresh rate further, to almost 76 Hz, by using the fact that monitors can often sync horizontally at 2khz or so higher than rated, and by lowering VFL somewhat (that is, taking less than 75% of 944 in the example above). But before you try this "overdriving" maneuver, if you do, make *sure* that your monitor electron guns can sync up to 76 Hz vertical. (the popular NEC 4D, for instance, cannot. It goes only up to 75 Hz VSF). (See Overdriving Your Monitor for more general discussion of

this issue.)

So far, most of this is simple arithmetic and basic facts about raster displays. Hardly any black magic at all!

XFree86 Video Timings HOWTO **:** *Computing Frame Sizes*
Previous: *Memory Requirements*
Next: *Black Magic and Sync Pulses*

9. Black Magic and Sync Pulses

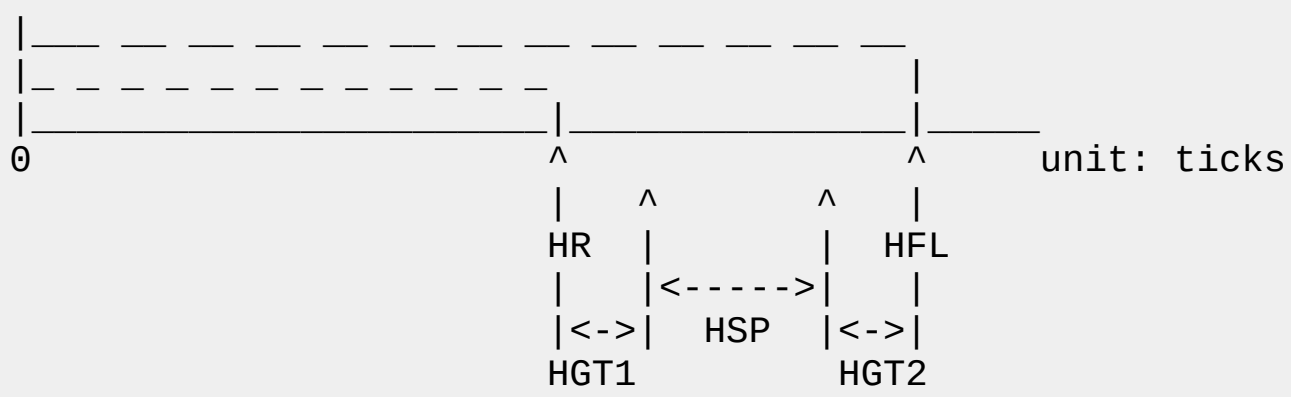
OK, now you've computed HFL/VFL numbers for your chosen dot clock, found the refresh rate acceptable, and checked that you have enough VRAM. Now for the real black magic -- you need to know when and where to place synchronization pulses.

The sync pulses actually control the horizontal and vertical scan frequencies of the monitor. The HSF and VSF you've pulled off the spec sheet are nominal, approximate maximum sync frequencies. The sync pulse in the signal from the adapter card tells the monitor how fast to actually run.

Recall the two pictures above? Only part of the time required for raster-scanning a frame is used for displaying viewable image (ie. your resolution).

9.1. Horizontal Sync:

By previous definition, it takes HFL ticks to trace the a horizontal scan line. Let's call the visible tick count (your horizontal screen resolution) HR. Then Obviously, HR < HFL by definition. For concreteness, let's assume both start at the same instant as shown below:



Now, we would like to place a sync pulse of length HSP as shown above, ie, between the end of clock ticks for display data and the end of clock ticks for the entire frame. Why so? because if we can achieve this, then your screen image won't shift to the right or to the left. It will be where it supposed to be on the screen, covering squarely the monitor's viewable area.

Furthermore, we want about 30 ticks of "guard time" on either side of the sync pulse. This is represented by HGT1 and HGT2. In a typical configuration HGT1 != HGT2, but if you're building a configuration from scratch, you want to start your experimentation with them equal (that is, with the sync pulse centered).

The symptom of a misplaced sync pulse is that the image is displaced on the screen, with one border

excessively wide and the other side of the image wrapped around the screen edge, producing a white edge line and a band of "ghost image" on that side. A way-out-of-place vertical sync pulse can actually cause the image to roll like a TV with a mis-adjusted vertical hold (in fact, it's the same phenomenon at work).

If you're lucky, your monitor's sync pulse widths will be documented on its specification page. If not, here's where the real black magic starts...

You'll have to do a little trial and error for this part. But most of the time, we can safely assume that a sync pulse is about 3.5 to 4.0 microsecond in length.

For concreteness again, let's take HSP to be 3.8 microseconds (which btw, is not a bad value to start with when experimenting).

Now, using the 65Mhz clock timing above, we know HSP is equivalent to 247 clock ticks (= 65 * 10**6 * 3.8 * 10^-6) [recall M=10^6, micro=10^-6]

Some makers like to quote their horizontal framing parameters as timings rather than dot widths. You may see the following terms:

active time (HAT)

Corresponds to HR, but in milliseconds. HAT * DCF = HR.

blanking time (HBT)

Corresponds to (HFL - HR), but in milliseconds. HBT * DCF = (HFL - HR).

front porch (HFP)

This is just HGT1.

sync time

This is just HSP.

back porch (HBP)

This is just HGT2.

9.2. Vertical Sync:

Going back to the picture above, how do we place the 247 clock ticks as shown in the picture?

Using our example, HR is 944 and HFL is 1176. The difference between the two is 1176 - 944=232 < 247! Obviously we have to do some adjustment here. What can we do?

The first thing is to raise 1176 to 1184, and lower 944 to 936. Now the difference = 1184-936= 248. Hmm, closer.

Next, instead using 3.8, we use 3.5 for calculating HSP; then, we have 65*3.5=227. Looks better. But 248 is not much higher than 227. It's normally necessary to have 30 or so clock ticks between HR and the start of SP, and the same for the end of SP and HFL. AND they have to be multiple of eight! Are we stuck?

No. Let's do this, 936 % 8 = 0, (936 + 32) % 8 = 0 too. But 936 + 32 = 968, 968 + 227 = 1195, 1195 + 32 = 1227. Hmm.. this looks not too bad. But it's not a multiple of 8, so let's round it up to 1232.

But now we have potential trouble, the sync pulse is no longer placed right in the middle between h and H any more. Happily, using our calculator we find 1232 - 32 = 1200 is also a multiple of 8 and (1232 - 32) - 968 = 232 corresponding using a sync pulse of 3.57 micro second long, still reasonable.

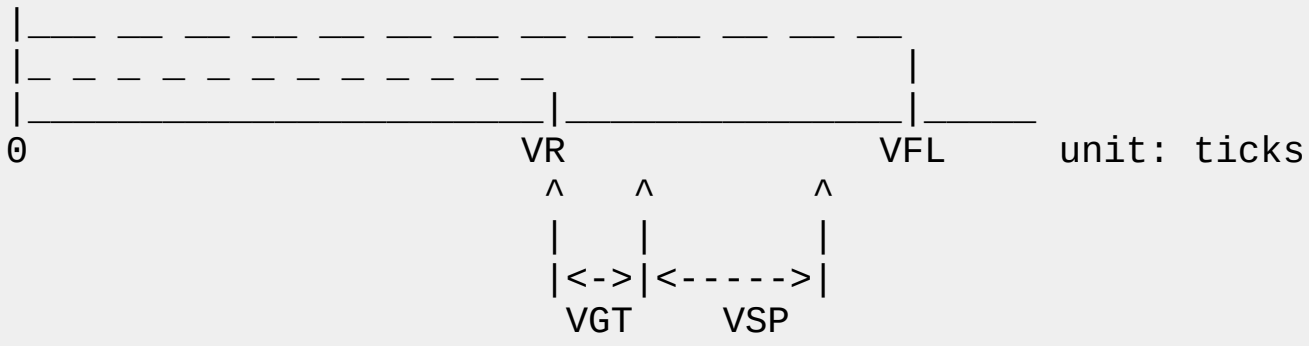
In addition, 936/1232 ~ 0.76 or 76%, still not far from 80%, so it should be all right.

Furthermore, using the current horizontal frame length, we basically ask our monitor to sync at 52.7khz (= 65Mhz/1232) which is within its capability. No problems.

Using rules of thumb we mentioned before, 936*75%=702, This is our new vertical resolution. 702 * 1.05 = 737, our new vertical frame length.

Screen refresh rate = 65Mhz/(737*1232)=71.6 Hz. This is still excellent.

Figuring the vertical sync pulse layout is similar:



We start the sync pulse just past the end of the vertical display data ticks. VGT is the vertical guard time required for the sync pulse. Most monitors are comfortable with a VGT of 0 (no guard time) and we'll use that in this example. A few need two or three ticks of guard time, and it usually doesn't hurt to add that.

Returning to the example: since by the definition of frame length, a vertical tick is the time for tracing a complete HORIZONTAL frame, therefore in our example, it is 1232/65Mhz=18.95us.

Experience shows that a vertical sync pulse should be in the range of 50us and 300us. As an example let's use 150us, which translates into 8 vertical clock ticks (150us/18.95us~8).

Some makers like to quote their vertical framing parameters as timings rather than dot widths. You may see the following terms:

active time (VAT)

Corresponds to VR, but in milliseconds. VAT * VSF = VR.

blanking time (VBT)

Corresponds to (VFL - VR), but in milliseconds. VBT * VSF = (VFL - VR).


TOC

front porch (VFP)

This is just VGT.

sync time

This is just VSP.

back porch (VBP)

This is like a second guard time after the vertical sync pulse. It is often zero.

XFree86 Video Timings HOWTO **:** *Black Magic and Sync Pulses*
Previous: *Computing Frame Sizes*
Next: *Putting it All Together*

10. Putting it All Together

The Xconfig file Table of Video Modes contains lines of numbers, with each line being a complete specification for one mode of X-server operation. The fields are grouped into four sections, the name section, the clock frequency section, the horizontal section, and the vertical section.

The name section contains one field, the name of the video mode specified by the rest of the line. This name is referred to on the "Modes" line of the Graphics Driver Setup section of the Xconfig file. The name field may be omitted if the name of a previous line is the same as the current line.

The dot clock section contains only the dot clock (what we've called DCF) field of the video mode line. The number in this field specifies what dot clock was used to generate the numbers in the following sections.

The horizontal section consists of four fields which specify how each horizontal line on the display is to be generated. The first field of the section contains the number of dots per line which will be illuminated to form the picture (what we've called HR). The second field of the section indicates at which dot the horizontal sync pulse will begin. The third field indicates at which dot the horizontal sync pulse will end. The fourth field specifies the total horizontal frame length (HFL).

The vertical section also contains four fields. The first field contains the number of visible lines which will appear on the display (VR). The second field indicates the line number at which the vertical sync pulse will begin. The third field specifies the line number at which the vertical sync pulse will end. The fourth field contains the total vertical frame length (VFL).

Example:

```
#Modename     clock  horizontal timing  vertical timing

"752x564"     40    752 784  944 1088  564 567 569 611
              44.5  752 792  976 1240  564 567 570 600
```

(Note: stock X11R5 doesn't support fractional dot clocks.)

For Xconfig, all of the numbers just mentioned - the number of illuminated dots on the line, the number of dots separating the illuminated dots from the beginning of the sync pulse, the number of dots representing the duration of the pulse, and the number of dots after the end of the sync pulse - are added to produce the number of dots per line. The number of horizontal dots must be evenly divisible by eight.

Example horizontal numbers: 800 864 1024 1088

This sample line has the number of illuminated dots (800) followed by the number of the dot when the sync pulse starts (864), followed by the number of the dot when the sync pulse ends (1024), followed by

the number of the last dot on the horizontal line (1088).

Note again that all of the horizontal numbers (800, 864, 1024, and 1088) are divisible by eight! This is not required of the vertical numbers.

The number of lines from the top of the display to the bottom form the frame. The basic timing signal for a frame is the line. A number of lines will contain the picture. After the last illuminated line has been displayed, a delay of a number of lines will occur before the vertical sync pulse is generated. Then the sync pulse will last for a few lines, and finally the last lines in the frame, the delay required after the pulse, will be generated. The numbers that specify this mode of operation are entered in a manner similar to the following example.

Example vertical numbers: 600 603 609 630

This example indicates that there are 600 visible lines on the display, that the vertical sync pulse starts with the 603rd line and ends with the 609th, and that there are 630 total lines being used.

Note that the vertical numbers don't have to be divisible by eight!

Let's return to the example we've been working. According to the above, all we need to do from now on is to write our result into Xconfig as follows:

```
    <name>   DCF      HR   SH1 SH2    HFL    VR   SV1 SV2 VFL
```

where SH1 is the start tick of the horizontal sync pulse and SH2 is its end tick; similarly, SV1 is the start tick of the vertical sync pulse and SV2 is its end tick.

```
    #name    clock    horizontal timing    vertical timing     flag
    936x702  65       936 968 1200 1232    702 702 710 737
```

No special flag necessary; this is a non-interlaced mode. Now we are really done.

XFree86 Video Timings HOWTO **:** *Putting it All Together*

Previous: *Black Magic and Sync Pulses*

Next: *Overdriving Your Monitor*

11. Overdriving Your Monitor

You should absolutely *not* try exceeding your monitor's scan rates if it's a fixed-frequency type. You can smoke your hardware doing this! There are potentially subtler problems with overdriving a multisync monitor which you should be aware of.

Having a pixel clock higher than the monitor's maximum bandwidth is rather harmless, in contrast. (Note: the theoretical limit of discernible features is reached when the pixel clock reaches double the monitor's bandwidth. This is a straightforward application of Nyquist's Theorem: consider the pixels as a spatially distributed series of samples of the drive signals and you'll see why.)

It's exceeding the rated maximum sync frequencies that's problematic. Some modern monitors might have protection circuitry that shuts the monitor down at dangerous scan rates, but don't rely on it. In particular there are older multisync monitors (like the Multisync II) which use just one horizontal transformer. These monitors will not have much protection against overdriving them. While you necessarily have high voltage regulation circuitry (which can be absent in fixed frequency monitors), it will not necessarily cover every conceivable frequency range, especially in cheaper models. This not only implies more wear on the circuitry, it can also cause the screen phosphors to age faster, and cause more than the specified radiation (including X-rays) to be emitted from the monitor.

Another importance of the bandwidth is that the monitor's input impedance is specified only for that range, and using higher frequencies can cause reflections probably causing minor screen interferences, and radio disturbance.

However, the basic problematic magnitude in question here is the slew rate (the steepness of the video signals) of the video output drivers, and that is usually independent of the actual pixel frequency, but (if your board manufacturer cares about such problems) related to the maximum pixel frequency of the board.

So be careful out there...

12. Using Interlaced Modes

(This section is largely due to David Kastrup <dak@pool.informatik.rwth-aachen.de>)

At a fixed dot clock, an interlaced display is going to have considerably less noticeable flicker than a non-interlaced display, if the vertical circuitry of your monitor is able to support it stably. It is because of this that interlaced modes were invented in the first place.

Interlaced modes got their bad repute because they are inferior to their non-interlaced companions at the same vertical scan frequency, VSF (which is what is usually given in advertisements). But they are definitely superior at the same horizontal scan rate, and that's where the decisive limits of your monitor/graphics card usually lie.

At a fixed *refresh rate* (or half frame rate, or VSF) the interlaced display will flicker more: a 90Hz interlaced display will be inferior to a 90Hz non-interlaced display. It will, however, need only half the video bandwidth and half the horizontal scan rate. If you compared it to a non-interlaced mode with the same dot clock and the same scan rates, it would be vastly superior: 45Hz non-interlaced is intolerable. With 90Hz interlaced, I have worked for years with my Multisync 3D (at 1024x768) and am very satisfied. I'd guess you'd need at least a 70Hz non-interlaced display for similar comfort.

You have to watch a few points, though: use interlaced modes only at high resolutions, so that the alternately lighted lines are close together. You might want to play with sync pulse widths and positions to get the most stable line positions. If alternating lines are bright and dark, interlace will *jump* at you. I have one application that chooses such a dot pattern for a menu background (XCept, no other application I know does that, fortunately). I switch to 800x600 for using XCept because it really hurts my eyes otherwise.

For the same reason, use at least 100dpi fonts, or other fonts where horizontal beams are at least two lines thick (for high resolutions, nothing else will make sense anyhow).

And of course, never use an interlaced mode when your hardware would support a non-interlaced one with similar refresh rate.

If, however, you find that for some resolution you are pushing either monitor or graphics card to their upper limits, and getting unsatisfactory flickery or washed out (bandwidth exceeded) display, you might want to try tackling the same resolution using an interlaced mode. Of course this is useless if the VSF of your monitor is already close to its limits.

Design of interlaced modes is easy: do it like a non-interlaced mode. Just two more considerations are necessary: you need an odd total number of vertical lines (the last number in your mode line), and when you specify the "interlace" flag, the actual vertical frame rate for your monitor doubles. Your monitor needs to support a 90Hz frame rate if the mode you specified looks like a 45Hz mode apart from the "Interlace" flag.

As an example, here is my modeline for 1024x768 interlaced: my Multisync 3D will support up to 90Hz vertical and 38kHz horizontal.


```
ModeLine "1024x768" 45 1024 1048 1208 1248 768 768 776 807 Inter
```

Both limits are pretty much exhausted with this mode. Specifying the same mode, just without the "Interlace" flag, still is almost at the limit of the monitor's horizontal capacity (and strictly speaking, a bit under the lower limit of vertical scan rate), but produces an intolerably flickery display.

Basic design rules: if you have designed a mode at less than half of your monitor's vertical capacity, make the vertical total of lines odd and add the "Interlace" flag. The display's quality should vastly improve in most cases.

If you have a non-interlaced mode otherwise exhausting your monitor's specs where the vertical scan rate lies about 30% or more under the maximum of your monitor, hand-designing an interlaced mode (probably with somewhat higher resolution) could deliver superior results, but I won't promise it.

XFree86 Video Timings HOWTO **:** *Using Interlaced Modes*
Previous: *Overdriving Your Monitor*
Next: *Questions and Answers*

13. Questions and Answers

Q. The example you gave is not a standard screen size, can I use it?

A. Why not? There is NO reason whatsoever why you have to use 640x480, 800x600, or even 1024x768. The XFree86 servers let you configure your hardware with a lot of freedom. It usually takes two to three tries to come up the right one. The important thing to shoot for is high refresh rate with reasonable viewing area. not high resolution at the price of eye-tearing flicker!

Q. It this the only resolution given the 65Mhz dot clock and 55Khz HSF?

A. Absolutely not! You are encouraged to follow the general procedure and do some trial-and-error to come up a setting that's really to your liking. Experimenting with this can be lots of fun. Most settings may just give you nasty video hash, but in practice a modern multi-sync monitor is usually not damaged easily. Be sure though, that your monitor can support the frame rates of your mode before using it for longer times.

Beware fixed-frequency monitors! This kind of hacking around can damage them rather quickly. Be sure you use valid refresh rates for *every* experiment on them.

Q. You just mentioned two standard resolutions. In Xconfig, there are many standard resolutions available, can you tell me whether there's any point in tinkering with timings?

A. Absolutely! Take, for example, the "standard" 640x480 listed in the current Xconfig. It employs 25Mhz driving frequency, frame lengths are 800 and 525 => refresh rate ~ 59.5Hz. Not too bad. But 28Mhz is a commonly available driving frequency from many SVGA boards. If we use it to drive 640x480, following the procedure we discussed above, you would get frame lengths like 812 and 505. Now the refresh rate is raised to 68Hz, a quite significant improvement over the standard one.

Q. Can you summarize what we have discussed so far?

A. In a nutshell:

1. for any fixed driving frequency, raising max resolution incurs the penalty of lowering refresh rate and thus introducing more flicker.
2. if high resolution is desirable and your monitor supports it, try to get a SVGA card that provides a matching dot clock or DCF. The higher, the better!

14. Fixing Problems with the Image.

OK, so you've got your X configuration numbers. You put them in Xconfig with a test mode label. You fire up X, hot-key to the new mode, ... and the image doesn't look right. What do you do? Here's a list of common problems and how to fix them.

(Fixing these minor distortions is where **xvidtune**(1) really shines.)

You *move* the image by changing the sync pulse timing. You *scale* it by changing the frame length (you need to move the sync pulse to keep it in the same relative position, otherwise scaling will move the image as well). Here are some more specific recipes:

The horizontal and vertical positions are independent. That is, moving the image horizontally doesn't affect placement vertically, or vice-versa. However, the same is not quite true of scaling. While changing the horizontal size does nothing to the vertical size or vice versa, the total change in both may be limited. In particular, if your image is too large in both dimensions you will probably have to go to a higher dot clock to fix it. Since this raises the usable resolution, it is seldom a problem!

14.1. The image is displaced to the left or right

To fix this, move the horizontal sync pulse. That is, increment or decrement (by a multiple of 8) the middle two numbers of the horizontal timing section that define the leading and trailing edge of the horizontal sync pulse.

If the image is shifted left (right border too large, you want to move the image to the right) decrement the numbers. If the image is shifted right (left border too large, you want it to move left) increment the sync pulse.

14.2. The image is displaced up or down

To fix this, move the vertical sync pulse. That is, increment or decrement the middle two numbers of the vertical timing section that define the leading and trailing edge of the vertical sync pulse.

If the image is shifted up (lower border too large, you want to move the image down) decrement the numbers. If the image is shifted down (top border too large, you want it to move up) increment the numbers.

14.3. The image is too large both horizontally and vertically

Switch to a higher card clock speed. If you have multiple modes in your clock file, possibly a lower-speed one is being activated by mistake.

14.4. The image is too wide (too narrow) horizontally

To fix this, increase (decrease) the horizontal frame length. That is, change the fourth number in the first timing section. To avoid moving the image, also move the sync pulse (second and third numbers) half as far, to keep it in the same relative position.

14.5. The image is too deep (too shallow) vertically

To fix this, increase (decrease) the vertical frame length. That is, change the fourth number in the second timing section. To avoid moving the image, also move the sync pulse (second and third numbers) half as far, to keep it in the same relative position.

Any distortion that can't be handled by combining these techniques is probably evidence of something more basically wrong, like a calculation mistake or a faster dot clock than the monitor can handle.

Finally, remember that increasing either frame length will decrease your refresh rate, and vice-versa.

TOC

15. Plotting Monitor Capabilities

To plot a monitor mode diagram, you'll need the gnuplot package (a freeware plotting language for UNIX-like operating systems) and the tool `modeplot`, a shell/gnuplot script to plot the diagram from your monitor characteristics, entered as command-line options.

Here is a copy of modeplot:

```
#!/bin/sh
#
# modeplot -- generate X mode plot of available monitor modes
#
# Do `modeplot -?' to see the control options.
#
# ($Id: video-modes.sgml,v 1.2 1997/08/08 15:07:24 esr Exp $)

# Monitor description. Bandwidth in MHz, horizontal frequencies in kHz
# and vertical frequencies in Hz.
TITLE="Viewsonic 21PS"
BANDWIDTH=185
MINHSF=31
MAXHSF=85
MINVSF=50
MAXVSF=160
ASPECT="4/3"
vesa=72.5       # VESA-recommended minimum refresh rate

while [ "$1" != "" ]
do
        case $1 in
        -t) TITLE="$2"; shift;;
        -b) BANDWIDTH="$2"; shift;;
        -h) MINHSF="$2" MAXHSF="$3"; shift; shift;;
        -v) MINVSF="$2" MAXVSF="$3"; shift; shift;;
        -a) ASPECT="$2"; shift;;
        -g) GNUOPTS="$2"; shift;;
        -?) cat <<EOF
modeplot control switches:

-t "<description>"      name of monitor             defaults to "Viewsonic 21PS"
-b <nn>                 bandwidth in MHz            defaults to 185
-h <min> <max>          min & max HSF (kHz)         defaults to 31 85
-v <min> <max>          min & max VSF (Hz)          defaults to 50 160
-a <aspect ratio>       aspect ratio                defaults to 4/3
-g "<options>"          pass options to gnuplot
```


```
The -b, -h and -v options are required, -a, -t, -g optional.  You can
use -g to pass a device type to gnuplot so that (for example) modeplot's
output can be redirected to a printer.  See gnuplot(1) for  details.

The modeplot tool was created by Eric S. Raymond <esr@thyrsus.com> based on
analysis and scratch code by Martin Lottermoser <Martin.Lottermoser@mch.sni.de>

This is modeplot $Revision: 1.2 $
EOF
                exit;;
        esac
        shift
done

gnuplot $GNUOPTS <<EOF
set title "$TITLE Mode Plot"

# Magic numbers.  Unfortunately, the plot is quite sensitive to changes in
# these, and they may fail to represent reality on some monitors.  We need
# to fix values to get even an approximation of the mode diagram.  These come
# from looking at lots of values in the ModeDB database.
F1 = 1.30       # multiplier to convert horizontal resolution to frame width
F2 = 1.05       # multiplier to convert vertical resolution to frame height

# Function definitions (multiplication by 1.0 forces real-number arithmetic)
ac = (1.0*$ASPECT)*F1/F2
refresh(hsync, dcf) = ac * (hsync**2)/(1.0*dcf)
dotclock(hsync, rr) = ac * (hsync**2)/(1.0*rr)
resolution(hv, dcf) = dcf * (10**6)/(hv * F1 * F2)

# Put labels on the axes
set xlabel 'DCF (MHz)'
set ylabel 'RR (Hz)' 6  # Put it right over the Y axis

# Generate diagram
set grid
set label "VB" at $BANDWIDTH+1, ($MAXVSF + $MINVSF) / 2 left
set arrow from $BANDWIDTH, $MINVSF to $BANDWIDTH, $MAXVSF nohead
set label "max VSF" at 1, $MAXVSF-1.5
set arrow from 0, $MAXVSF to $BANDWIDTH, $MAXVSF nohead
set label "min VSF" at 1, $MINVSF-1.5
set arrow from 0, $MINVSF to $BANDWIDTH, $MINVSF nohead
set label "min HSF" at dotclock($MINHSF, $MAXVSF+17), $MAXVSF + 17 right
set label "max HSF" at dotclock($MAXHSF, $MAXVSF+17), $MAXVSF + 17 right
set label "VESA $vesa" at 1, $vesa-1.5
set arrow from 0, $vesa to $BANDWIDTH, $vesa nohead # style -1
plot [dcf=0:1.1*$BANDWIDTH] [$MINVSF-10:$MAXVSF+20] \
  refresh($MINHSF, dcf) notitle with lines 1, \
  refresh($MAXHSF, dcf) notitle with lines 1, \
  resolution(640*480,   dcf) title "640x480  " with points 2, \
```


```
  resolution(800*600,   dcf) title "800x600  " with points 3, \
  resolution(1024*768,  dcf) title "1024x768 " with points 4, \
  resolution(1280*1024, dcf) title "1280x1024" with points 5, \
  resolution(1600*1280, dcf) title "1600x1200" with points 6

pause 9999
EOF
```

Once you know you have `modeplot` and the gnuplot package in place, you'll need the following monitor characteristics:

- video bandwidth (VB)
- range of horizontal sync frequency (HSF)
- range of vertical sync frequency (VSF)

The plot program needs to make some simplifying assumptions which are not necessarily correct. This is the reason why the resulting diagram is only a rough description. These assumptions are:

1. All resolutions have a single fixed aspect ratio AR = HR/VR. Standard resolutions have AR = 4/3 or AR = 5/4. The `modeplot` programs assumes 4/3 by default, but you can override this.
2. For the modes considered, horizontal and vertical frame lengths are fixed multiples of horizontal and vertical resolutions, respectively:

```
HFL = F1 * HR
VFL = F2 * VR
```

As a rough guide, take F1 = 1.30 and F2 = 1.05 (see frame "Computing Frame Sizes").

Now take a particular sync frequency, HSF. Given the assumptions just presented, every value for the clock rate DCF already determines the refresh rate RR, i.e. for every value of HSF there is a function RR(DCF). This can be derived as follows.

The refresh rate is equal to the clock rate divided by the product of the frame sizes:

```
RR = DCF / (HFL * VFL)          (*)
```

On the other hand, the horizontal frame length is equal to the clock rate divided by the horizontal sync frequency:

```
HFL = DCF / HSF                 (**)
```

VFL can be reduced to HFL be means of the two assumptions above:

```
VFL = F2 * VR
    = F2 * (HR / AR)
    = (F2/F1) * HFL / AR        (***)
```

Inserting (**) and (***) into (*) we obtain:

```
RR = DCF / ((F2/F1) * HFL**2 / AR)
   = (F1/F2) * AR * DCF * (HSF/DCF)**2
   = (F1/F2) * AR * HSF**2 / DCF
```

For fixed HSF, F1, F2 and AR, this is a hyperbola in our diagram. Drawing two such curves for minimum and

maximum horizontal sync frequencies we have obtained the two remaining boundaries of the permitted region.

The straight lines crossing the capability region represent particular resolutions. This is based on (*) and the second assumption:

```
RR = DCF / (HFL * VFL) = DCF / (F1 * HR * F2 * VR)
```

By drawing such lines for all resolutions one is interested in, one can immediately read off the possible relations between resolution, clock rate and refresh rate of which the monitor is capable. Note that these lines do not depend on monitor properties, but they do depend on the second assumption.

The `modeplot` tool provides you with an easy way to do this. Do `modeplot -?` to see its control options. A typical invocation looks like this:

```
modeplot -t "Swan SW617" -b 85 -v 50 90 -h 31 58
```

The -b option specifies video bandwidth; -v and -h set horizontal and vertical sync frequency ranges.

When reading the output of `modeplot`, always bear in mind that it gives only an approximate description. For example, it disregards limitations on HFL resulting from a minimum required sync pulse width, and it can only be accurate as far as the assumptions are. It is therefore no substitute for a detailed calculation (involving some black magic) as presented in Putting it All Together. However, it should give you a better feeling for what is possible and which tradeoffs are involved.

16. Credits

The original ancestor of this document was by Chin Fang <fangchin@leland.stanford.edu>.

Eric S. Raymond <esr@snark.thyrsus.com> reworked, reorganized, and massively rewrote Chin Fang's original in an attempt to understand it. In the process, he merged in most of a different how-to by Bob Crosson <crosson@cam.nist.gov>.

The material on interlaced modes is largely by David Kastrup <dak@pool.informatik.rwth-aachen.de>

Martin Lottermoser <Martin.Lottermoser@mch.sni.de> contributed the idea of using gnuplot to make mode diagrams and did the mathematical analysis behind `modeplot`. The distributed `modeplot` was redesigned and generalized by ESR from Martin's original gnuplot code for one case.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/VidModes.sgml,v 3.11.2.2 1998/02/20
23:10:30 dawes Exp $
```

```
$XConsortium: VidModes.sgml /main/7 1996/02/21 17:46:17 kaleb $
```

How to add an (S)VGA driver to XFree8

Copyright (c) 1993, 1994 David E. Wexelblat <dwex@XFree86.org>

Issue 1.3 - May 29, 1994

1. Introduction
2. Getting Started
3. Directory Tree Structure
4. Setting Up The Build Information
5. The Bank-Switching Functions
6. The Driver Itself
    - 6.1. Multiple Chipsets And Options
    - 6.2. Data Structures
    - 6.3. The Ident() function
    - 6.4. The ClockSelect() function
    - 6.5. The Probe() function
    - 6.6. The EnterLeave() function
    - 6.7. The Restore() function
    - 6.8. The Save() function
    - 6.9. The Init() function
    - 6.10. The Adjust() function
    - 6.11. The ValidMode() function
    - 6.12. The SaveScreen() function
    - 6.13. The GetMode() function
    - 6.14. The FbInit() function

TOC

7. Building The New Server

8. Debugging

9. Advice

10. Advanced Topics

11. References

12. Vendor Contact Information

1. Introduction

Adding support for a new SVGA chipset to XFree86 is a challenging project for someone who wants to learn more about hardware-level programming. It can be fraught with hazards (in particular, crashing the machine is all too common). But in the end, when the server comes up and functions, it is immensely satisfying.

Adding support for an SVGA chipset does not change any of the basic functioning of the server. It is still a dumb 8-bit PseudoColor server or 1-bit StaticGray server. Adding support for new hardware (e.g. accelerated chips) is a major undertaking, and is not anywhere near formalized enough yet that it can be documented.

Nonetheless, the driver-level programming here is a good introduction. And can well be the first step for adding support for an accelerated chipset, as many are SVGA-supersets. Writing an SVGA-level driver for the chipset can provide a stable development platform for making use of new features (in fact, this has been done for the S3, Cirrus, and WD accelerated chipsets, for internal use as the accelerated servers are developed for XFree86 2.0).

Now let's get down to it. In addition to this documentation, a stub driver has been provided. This should provide you a complete framework for your new driver. Don't let the size of this document persuade you that this is an overly difficult task. A lot of work has been put into making this document as close to complete as possible; hence it should, in theory, be possible to use this as a cookbook, and come out with a working driver when you reach the end. I do advise that you read it all the way through before starting.

2. Getting Started

The first step in developing a new driver is to get the documentation for your chipset. I've included a list of vendor contact information that I have collected so far (it's far from complete, so if you have any that isn't on the list, please send it to me). You need to obtain the databook for the chipset. Make sure that the person you speak to is aware that you intend to do register-level programming (so they don't send you the EE-style datasheet). Ask for any example code, or developer's kits, etc. I've learned that at the SVGA level, in general, a databook that lists and describes the registers is the most you can hope to find.

If you are not familiar with VGA register-level programming, you should get (and read!) a copy of Richard Ferraro's bible (see references below). The best way to understand what is happening in the server is to study the workings of the monochrome server's ``generic'' server, and compare it with the documentation in Ferraro's book (be aware that there are a few errors in the book). You can find the generic-VGA-register handling functions in the file ``vgaHW.c''.

Once you understand what's happening in the generic server, you should study one or more of the existing SVGA drivers. Obtain the databook for a supported SVGA chipset, and study the documentation along with the code. When you have a good understanding of what that driver does over and above the generic VGA, you will know what information you need to obtain from the databook for the new chipset. Once you have this information, you are ready to begin work on your new driver.

3. Directory Tree Structure

Here is an outline of the directory tree structure for the source tree. Only directories/files that are relevant to writing a driver are presented. The structure for the Link Kit is presented below.

xc/config/cf/

site.def

Local configuration customization

xf86site.def

XFree86 local configuration customization

xc/programs/Xserver/hw/xfree86/

The server source

common/

Files common to all of the server (XF86Config parser, I/O device handlers, etc)

xf86.h

Contains the `ScrnInfoRec' data structure

xf86_Option.h

Contains option flags

compiler.h

Contains in-line assembler macros and utility functions

os-support/

OS-support layer

assyntax.h

Contains macro-ized assembler mnemonics

xf86_OSlib.h

OS-support includes, defines, and prototypes

LinkKit/

site.def.LK

Template for Link Kit site.def

vga256/

256-color VGA server directories

vga/

The generic VGA handling code

vga.h

Contains the `vgaVideoChipRec' and `vgaHWRec' data structures

vgaHW.c

Contains the generic-VGA-register handling functions **vgaHWInit()**, **vgaHWSave()** and **vgaHWRestore()**.

drivers/

Contains the SVGA driver subdirectories. Each contains an Imakefile, a .c file for the driver, and a .s file for the bank- switching functions.

vga2/

The monochrome vga server directories. Most of the files are linked from vga256, and the differences handled by conditional compilation.

drivers/

The SVGA driver subdirectories. The `generic' VGA driver is also located here.

vga16/

The 16-color vga server directories. Most of the files are linked from vga256, and the differences handled by conditional compilation.

drivers/

The SVGA driver subdirectories.

VGADriverDoc/

This documentation and the stub driver.

The Link Kit is usually installed in /usr/X11R6/lib/Server. The Link Kit contains everything that is needed to relink the server. It is possible to write a new driver and build a new server without having even the server source installed.

Server/

site.def

Local configuration customization

include/

All of the include files listed under the `common' directory above

drivers/

All of the SVGA drivers

vga2/

TOC

The SVGA driver subdirectories.

vga16/

The SVGA driver subdirectories.

vga256/

The SVGA driver subdirectories.

VGADriverDoc/

The directory with this documentation and the stub driver. `vgaHW.c' is also copied here, for reference (it is not built as part of the Link Kit).

How to add an (S)VGA driver to XFree86 **:** *Directory Tree Structure*
Previous: *Getting Started*
Next: *Setting Up The Build Information*

4. Setting Up The Build Information

This section describes the peripheral configuration and build steps that must be performed to set up for your new driver. The steps are the same whether you are building from the source tree of from the Link Kit; only the locations of the files is different. Here are the configuration steps that must be followed:

1. Choose the name for your driver subdirectory and data structures. Since the current driver scheme allows (in fact, encourages) putting drivers for multiple related chipsets in a single driver, it is usually best to use the vendor name, rather than a chipset version. The fact that older XFree86 drivers do not follow this convention should not deter you from using it now - most of that code was developed before the driver interface had been made flexible and extensible. For this documentation, we'll use chips from the SuperDuper Chips vendor. Hence, we'll use `sdc' for the name of the driver.
2. Decide whether your driver will support the color server, the monochrome server, or both. For this documentation, we will assume that both the color and monochrome servers will be supported. If you intend to support only the color server, the steps for the monochrome server can be ignored. If you intend to support only the monochrome server, the steps for the color server listed should be performed for the monochrome server, and the monochrome steps ignored. Most of the existing drivers support only the color or both servers; the ``generic'' driver is the only driver (currently) that supports just the monochrome server.
3. Create your driver directories:
   - If you are working in the source tree, create the following directories:

     ```
     xc/programs/Xserver/hw/xfree86/vga256/drivers/sdc
     xc/programs/Xserver/hw/xfree86/vga16/drivers/sdc
     xc/programs/Xserver/hw/xfree86/vga2/drivers/sdc
     ```

   - If you are working in the Link Kit, create the following directories:

     ```
     /usr/X11R6/lib/Server/drivers/vga256/sdc
     /usr/X11R6/lib/Server/drivers/vga16/sdc
     /usr/X11R6/lib/Server/drivers/vga2/sdc
     ```

4. Set up the Imakefile parameters to cause your driver to be built:
   - If you are working in the source tree:
     1. Edit the file xc/config/cf/xfree86.cf, and add `sdc' to the list for the definitions for `XF86Vga256Drivers', `XF86Vga16Drivers' and `XF86Vga2Drivers'. You should put `sdc' just before `generic' in the list (i.e. second last), to ensure that none of the other driver's probe functions incorrectly detect the `sdc' chipset .
     2. Edit the file xc/config/cf/xf86site.def, and add the same entries in this file (this is just

a comment that shows the default values).

3. Edit the site.def.LK file in xc/programs/Xserver/hw/xfree86/LinkKit/, and add the same entries in this file. This is the prototype `site.def' file that will be installed in the Link Kit.

- If you are working in the Link Kit, edit the file /usr/X11R6/lib/Server/site.def, and add `sdc' to the `XF86Vga256Drivers', `XF86Vga16Drivers' and `XF86Vga2Drivers' definitions as described in (a) above.

5. Now copy the prototype files into your new directories:
   - If you are working in the source tree, copy the `stub' files as follows (directories are below xc/programs/Xserver):

     Imakefile.stub =>

     hw/xfree86/vga256/drivers/sdc/Imakefile

     stub_driver.c =>

     hw/xfree86/vga256/drivers/sdc/sdc_driver.c

     stub_bank.s =>

     hw/xfree86/vga256/drivers/sdc/sdc_bank.s

     Imakefile.stub =>

     hw/xfree86/vga16/drivers/sdc/Imakefile (then edit this Imakefile and make the changes described in the comments).

     Imakefile.stub =>

     hw/xfree86/vga2/drivers/sdc/Imakefile (then edit this Imakefile and make the changes described in the comments).

   - If you are working in the Link Kit, copy the `stub' files as follows:

     Imakefile.stub =>

     /usr/X11R6/lib/Server/drivers/vga256/sdc/Imakefile

     stub_driver.c =>

     /usr/X11R6/lib/Server/drivers/vga256/sdc/sdc_driver.c

     stub_bank.s =>

     /usr/X11R6/lib/Server/drivers/vga256/sdc/sdc_bank.s

     Imakefile.stub =>

     /usr/X11R6/lib/Server/drivers/vga16/sdc/Imakefile (then edit this Imakefile and make the changes described in the comments).

     Imakefile.stub =>

     /usr/X11R6/lib/Server/drivers/vga2/sdc/Imakefile (then edit this Imakefile and make the changes described in the comments).

6. Edit each of the files you've just copied, and replace `stub' with `sdc' and `STUB' with `SDC' wherever they appear.

That's all the prep work needed. Now it's time to work on the actual driver.

TOC

5. The Bank-Switching Functions

The normal VGA memory map is 64k starting at address 0xA0000. To access more than 64k of memory, SuperVGA chipsets implement ``bank switching'' - the high-order address bits are used to select the bank of memory in which operations will take place. The size and number of these banks varies, and will be spelled out in the chipset documentation. A chipset will have zero, one or two bank registers. Likely the ONLY case of zero bank registers is a generic VGA, and hence is not a concern.

Note that some of the newer chipsets (e.g. Trident 8900CL, Cirrus) allow for a linear mapping of the video memory. While using such a scheme would improve the performance of the server, it is not currently supported. Hence there is no way to use such features for a new chipset.

Most SVGA chipsets have two bank registers. This is the most desirable structure (if any banking structure can be called ``desirable''), because data can be moved from one area of the screen to another with a simple `mov' instruction. There are two forms of dual-banking - one where the two bank operations define a read-only bank and a write-only bank, and one with two read/write windows. With the first form, the entire SVGA memory window is used for both read a write operations, and the two bank registers determine which bank is actually used (e.g. ET3000, ET4000). With the second form, the SVGA memory window is split into two read/write banks, with each bank pointer being used to control one window. In this case, one window is used for read operations and the other for write operations (e.g. PVGA1/Western Digital, Cirrus).

A chipset that has a single bank register uses that one bank for both read and write access. This is problematic, because copying information from one part of the screen to another requires that the data be read in, stored, and then written out. Fortunately, the server is able to handle both one-bank and two-bank chipsets; the determination of behavior is defined by an entry in the driver data structure described below.

A driver requires that three assembly-language functions be written, in the file `sdc_bank.s'. These functions set the read bank - **SDCSetRead()**, the write bank - **SDCSetWrite()**, and set both banks - **SDCSetReadWrite()**. For a chipset with only one bank, all three will be declared as entry points to the same function (see the ``tvga8900'' driver for an example).

The functions are fairly simple - the bank number is passed to the function in register %al. The function will shift, bitmask, etc - whatever is required to put the bank number into the correct form - and then write it to the correct I/O port. For chipsets where the two banks are read-only HERE and write-only, the **SetReadWrite()** function will have to do this twice - once for each bank. For chipsets with two independent read/write windows, the **SetReadWrite()** function should use the same bank as the **SetWrite()** function.

A special note - these functions MUST be written in the macroized assembler format defined in the header file ``assyntax.h''. This will ensure that the correct assembler code will be generated, regardless of

OS. This macroized format currently supports USL, GNU, and Intel assembler formats.

That's all there is to the banking functions. Usually the chipset reference will give examples of this code; if not, it is not difficult to figure out, especially using the other drivers as examples.

---

6. The Driver Itself

Now it's time to get down to the real work - writing the major driver functions in the files sdc_driver.c. First, an overview of what the responsibilities of the driver are:

1. Provide a chipset-descriptor data structure to the server. This data structure contains pointers to the driver functions and some data-structure initialization as well.
2. Provide a driver-local data structure to hold the contents of the chipset registers. This data structure will contain a generic part and a driver-specific part. It is used to save the initial chipset state, and is initialized by the driver to put the chipset into different modes.
3. Provide an identification function that the server will call to list the chipsets that the driver is capable of supporting.
4. Provide a probe function that will identify this chipset as different from all others, and return a positive response if the chipset this driver supports is installed, and a negative response otherwise.
5. Provide a function to select dot-clocks available on the board.
6. Provide functions to save, restore, and initialize the driver- local data structure.
7. Provide a function to set the starting address for display in the video memory. This implements the virtual-screen for the server.
8. Perhaps provide a function for use during VT-switching.
9. Perhaps provide a function to check if each mode is suitable for the chipset being used.

Before stepping through the driver file in detail, here are some important issues:

1. If your driver supports both the color and monochrome servers, you should take care of both cases in the same file. Most things are the same - you can differentiate between the two with the MONOVGA `#define`. If the 16 color server is supported, code specific to it can be enabled with the XF86VGA16 `#define`. In most cases it is sufficient to put the following near the top of the stub_driver.c file:

```
#ifdef XF86VGA16
#define MONOVGA
#endif
```

2. The color server uses the SVGA's 8-bit packed-pixel mode. The monochrome and vga16 servers uses the VGA's 16-color mode (4 bit-planes). Only one plane is enabled for the monochrome server.
3. It is possible for you to define your monochrome driver so that no bank-switching is done. This is not particularly desirable, as it yields only 64k of viewing area.

Keeping these things in mind, you need to find the registers from your SVGA chipset that control the desired features. In particular, registers that control:

1. Clock select bits. The two low-order bits are part of the standard Miscellaneous Output Regist most SVGA chipsets will include 1 or 2 more bits, allowing the use of 8 or 16 discrete clocks.
2. Bank selection. The SVGA chipset will have one or two registers that control read/write bank selection.
3. CRTC extensions. The standard VGA registers don't have enough bits to address large displays. So the SVGA chipsets have extension bits.
4. Interlaced mode. Standard VGA does not support interlaced displays. So the SVGA chipset will have a bit somewhere to control interlaced mode. Some chipsets require additional registers to be set up to control interlaced mode
5. Starting address. The standard VGA only has 16 bits in which to specify the starting address for the display. This restricts the screen size usable by the virtual screen feature. The SVGA chipset will usually provide one or more extension bits.
6. Lock registers. Many SVGA chipset prevent modification of extended registers unless the registers are first ``unlocked''. You will need to disable protection of any registers you will need for other purposes.
7. Any other facilities. Some chipset may, for example, require that certain bits be set before you can access extended VGA memory (beyond the IBM-standard 256k). Or other facilities; read through all of the extended register descriptions and see if anything important leaps out at you.

If you are fortunate, the chipset vendor will include in the databook some tables of register settings for various BIOS modes. You can learn a lot about what manipulations you must do by looking at the various BIOS modes.

6.1. Multiple Chipsets And Options

It is possible, and in fact desirable, to have a single driver support multiple chipsets from the same vendor. If there are multiple supported chipsets, then you would have a series of #define's for them, and a variable `SDCchipset', which would be used throughout the driver when distinctions must be made. See the Trident and PVGA1/WD drivers for examples (the Tseng ET3000 and ET4000 are counter-examples - these were implemented before the driver interface allowed for multiple chipsets, so this example should NOT be followed). Note that you should only distinguish versions when your driver needs to do things differently for them. For example, suppose the SDC driver supports the SDC-1a, SDC-1b, and SDC-2 chipsets. The -1a and -1b are essentially the same, but different from the -2 chipset. Your driver should support the -1 and -2 chipsets, and not distinguish between the -1a and -1b. This will simplify things for the end user.

In cases where you want to give the user control of driver behavior, or there are things that cannot be determined without user intervention, you should use ``option'' flags. Say that board vendors that use the SDC chipsets have the option of providing 8 or 16 clocks. There's no way you can determine this from the chipset probe, so you provide an option flag to let the user select the behavior from the XF86Config file. The option flags are defined in the file ``xf86_option.h''. You should look to see if there is already a flag that can be reused. If so, use it in your driver. If not, add a new #define, and define the string->symbol mapping in the table in that file. To see how option flags are used, look at the ET4000, PVGA1/WD, and Trident drivers.

6.2. Data Structures

Once you have an understanding of what is needed from the above description, it is time to fill in the driver data structures. First we will deal with the `vgaSDCRec' structure. This data structure is the driver-local structure that holds the SVGA state information. The first entry in this data structure is ALWAYS `vgaHWRec std'. This piece holds the generic VGA portion of the information. After that, you will have one `unsigned char' field for each register that will be manipulated by your driver. That's all there is to this data structure.

Next you must initialize the `SDC' structure (type `vgaVideoChipRec'). This is the global structure that identifies your driver to the server. Its name MUST be `SDC', in all caps - i.e. it must match the directory name for your driver. This is required so that the Link Kit reconfiguration can identify all of the requisite directories and global data structures.

The first section of this structure simply holds pointers to the driver functions.

Next, you must initialize the information about how your chipset does bank switching. The following fields must be filled in:

1. ChipMapSize - the amount of memory that must be mapped into the server's address space. This is almost always 64k (from 0xA0000 to 0xAFFFF). Some chipsets use a 128k map (from 0xA0000 to 0xBFFFF). If your chipset gives an option, use the 64k window, as a 128k window rules out using a Hercules or Monochrome Display Adapter card with the SVGA.
2. ChipSegmentSize - the size of each bank within the ChipMapSize window. This is usually also 64k, however, some chipsets split the mapped window into a read portion and a write portion (for example the PVGA1/Western Digital chipsets).
3. ChipSegmentShift - the number of bits by which an address will be shifted right to mask of the bank number. This is log-base-2 of ChipSegmentSize.
4. ChipSegmentMask - a bitmask used to mask off the address within a given bank. This is (ChipSegmentSize-1).
5. ChipReadBottom,ChipReadTop - the addresses within the mapped window in which read operations can be done. Usually 0, and 64k, respectively, except for those chipset that have separate read and write windows.
6. ChipWriteBottom,ChipWriteTop - same as above, for write operations.
7. ChipUse2Banks - a boolean value for whether this chipset has one or two bank registers. This is used to set up the screen-to-screen operations properly.

There are three more fields that must be filled in:

1. ChipInterlaceType - this is either VGA_NO_DIVIDE_VERT or VGA_DIVIDE_VERT. Some chipsets require that the vertical timing numbers be divided in half for interlaced modes. Setting this flag will take care of that.
2. ChipOptionFlags - this should always be `{0,}' in the data structure initialization. This is a bitfield that contains the Option flags that are valid for this driver. The appropriate bits are initialized at the end of the Probe function.
3. ChipRounding - this gets set to the multiple by which the virtual width of the display must be rounded for the 256-color server. This value is usually 8, but may be 4 or 16 for some chipsets.


TOC

6.3. The Ident() function

The **Ident()** function is a very simple function. The server will call this function repeatedly, until a NULL is returned, when printing out the list of configured drivers. The **Ident()** function should return a chipset name for a supported chipset. The function is passed a number which increments from 0 on each iteration.

6.4. The ClockSelect() function

The **ClockSelect()** function is used during clock probing (i.e. when no `Clocks' line is specified in the XF86Config file) to select the dot-clock indicated by the number passed in the parameter. The function should set the chipset's clock-select bits according to the passed-in number. Two dummy values will be passed in as well (CLK_REG_SAVE, CLK_SAVE_RESTORE). When CLK_REG_SAVE is passed, the function should save away copies of any registers that will be modified during clock selection. When CLK_REG_RESTORE is passed, the function should restore these registers. This ensure that the clock-probing cannot corrupt registers.

This function should return FALSE if the passed-in index value is invalid or if the clock can't be set for some reason.

6.5. The Probe() function

The **Probe()** function is perhaps the most important, and perhaps the least intuitive function in the driver. The Probe function is required to identify the chipset independent of all other chipsets. If the user has specified a `Chipset' line in the XF86Config file, this is a simple string comparison check. Otherwise, you must use some other technique to figure out what chipset is installed. If you are lucky, the chipset will have an identification mechanism (ident/version registers, etc), and this will be documented in the databook. Otherwise, you will have to determine some scheme, using the reference materials listed below.

The identification is often done by looking for particular patterns in register, or for the existence of certain extended registers. Or with some boards/chipsets, the requisite information can be obtained by reading the BIOS for certain signature strings. The best advise is to study the existing probe functions, and use the reference documentation. You must be certain that your probe is non-destructive - if you modify a register, it must be saved before, and restored after.

Once the chipset is successfully identified, the **Probe()** function must do some other initializations:

1. If the user has not specified the `VideoRam' parameter in the XF86Config file, the amount of installed memory must be determined.
2. If the user has not specified the `Clocks' parameter in the XF86Config file, the values for the available dot-clocks must be determined. This is done by calling the **vgaGetClocks()** function, and passing it the number of clocks available and a pointer to the **ClockSelect()** function.
3. It is recommended that the `maxClock' field of the server's `vga256InfoRec' structure be filled in with the maximum dot-clock rate allowed for this chipset (specified in KHz). If this is not filled in a probe time, a default (currently 90MHz) will be used.

4. The `chipset' field of the server's `vga256InfoRec' structure must be initialized to the name of installed chipset.
5. If the driver will be used with the monochrome server, the `bankedMono' field of the server's `vga256InfoRec' structure must be set to indicate whether the monochrome driver supports banking.
6. If any option flags are used by this driver, the `ChipOptionFlags' structure in the `vgaVideoChipRec' must be initialized with the allowed option flags using the **OFLG_SET()** macro.

6.6. The EnterLeave() function

The **EnterLeave()** function is called whenever the virtual console on which the server runs is entered or left (for OSs without virtual consoles, the function is called when the server starts and again when it exits). The purpose of this function is to enable and disable I/O permissions (for OSs where such is required), and to unlock and relock access to ``protected'' registers that the driver must manipulate. It is a fairly trivial function, and can be implemented by following the comments in the stub driver.

6.7. The Restore() function

The **Restore()** function is used for restoring a saved video state. Note that `restore' is a bit of a misnomer - this function is used to both restore a saved state and to install a new one created by the server. The **Restore()** function must complete the following actions:

1. Ensure that Bank 0 is selected, and that any other state information required prior to writing out a new state has been set up.
2. Call **vgaHWRestore()** to restore the generic VGA portion of the state information. This function is in the vgaHW.c file.
3. Restore the chipset-specific portion of the state information. This may be done by simply writing out the register, or by doing a read/modify/write cycle if only certain bits are to be modified. Be sure to note the comment in the sample driver about how to handle clock-select bits.

6.8. The Save() function

The **Save()** function is used to extract the initial video state information when the server starts. The **Save()** function must complete the following actions:

1. Ensure that Bank 0 is selected.
2. Call **vgaHWSave()** to extract the generic VGA portion of the state information. This function is in the vgaHW.c file.
3. Extract the chipset-specific portion of the state information.

6.9. The Init() function

The **Init()** function is the second most important function in the driver (after the **Probe()** function). It is used to initialize a data structure for each of the defined display modes in the server. This function is required to initialize the entire `vgaSDCRec' data structure with the information needed to put the SVGA


TOC

chipset into the required state. The generic VGA portion of the structure is initialized with a call to **vgaHWInit()** (also located in vgaHW.c).

Once the generic portion is initialized, the **Init()** function can override any of the generic register initialization, if necessary. All of the other fields are filled in with the correct initialization. The information about the particular mode being initialized is passed in the `mode' parameter, a pointer to a `DisplayModeRec' structure. This can be dereferenced to determine the needed parameters.

If you only know how to initialize certain bits of the register, do that here, and make sure that the **Restore()** function does a read/modify/write to only manipulate those bits. Again, refer to the existing drivers for examples of what happens in this function.

6.10. The Adjust() function

The **Adjust()** function is another fairly basic function. It is called whenever the server needs to adjust the start of the displayed part of the video memory, due to scrolling of the virtual screen or when changing the displayed resolution. All it does is set the starting address on the chipset to match the specified coordinate. Follow the comments in the stub driver for details on how to implement it.

6.11. The ValidMode() function

The **ValidMode()** function is required. It is used to check for any chipset-dependent reasons why a graphics mode might not be valid. It gets called by higher levels of the code after the Probe() stage. In many cases no special checking will be required and this function will simply return TRUE always.

6.12. The SaveScreen() function

The **SaveScreen()** function is not needed by most chipsets. This function would only be required if the extended registers that your driver needs will be modified when a synchronous reset is performed on the SVGA chipset (your databook should tell you this). If you do NOT need this function, simply don't define it, and put `NoopDDA' in its place in the vgaVideoChipRec structure initialization (NoopDDA is a generic-use empty function).

If you DO need this function, it is fairly simple to do. It will be called twice - once before the reset, and again after. It will be passed a parameter of SS_START in the former case, and SS_FINISH in the latter. All that needs to be done is to save any registers that will be affected by the reset into static variables on the SS_START call, and then restore them on the SS_FINISH call.

6.13. The GetMode() function

The **GetMode()** function is not used as of XFree86 1.3; its place in the vgaVideoChipRec should be initialized to `NoopDDA'.

At some point in the future, this function will be used to enable the server and/or a standalone program using the server's driver libraries to do interactive video mode adjustments. This function will read the SVGA registers and fill in a DisplayModeRec structure with the current video mode.


TOC

6.14. The FbInit() function

The **FbInit()** function is required for drivers with accelerated graphics support. It is used to replace default cfb.banked functions with accelerated chip-specific versions. vga256LowlevFuncs is a struct containing a list of functions which can be replaced. This struct defined in vga256.h. Examples of **FbInit()** functions can be found in the et4000, pvga1 and cirrus drivers.

If you do NOT need this function, simply don't define it, and put `NoopDDA' in its place in the vgaVideoChipRec structure initialization.

How to add an (S)VGA driver to XFree86 **:** *The Driver Itself*
Previous: *The Bank-Switching Functions*
Next: *Building The New Server*

7. Building The New Server

As in the setup work, the steps for building the server depend whether you are working in the source tree or in the Link Kit. Here are the steps for the initial build after installing your new driver files:

- If you are working in the source tree, follow these steps: Go to xc/programs/Xserver, and enter `make Makefile', then `make Makefiles depend all'
- If you are working in the Link Kit, follow these steps:
  1. Go to /usr/X11R6/lib/Server, and enter `./mkmf'
  2. In the same directory, enter `make'

To rebuild the server after the initial build (e.g. after making changes to your driver):

- If you are working in the source tree, follow these steps:
  1. Go to the appropriate drivers/ directory (e.g., xc/programs/Xserver/hw/xfree86/vga256/drivers), and enter `make'.
  2. Go to xc/programs/Xserver, and enter `make loadXF86_SVGA' (to link the color server), `make loadXF86_VGA16' (to link the 16 color server) or `make loadXF86_Mono' (to link the mono server).
- If you are working in the Link Kit, follow these steps:
  1. Go to the appropriate driver directory, and enter `make'.
  2. Go to /usr/X11R6/lib/server, and enter `make loadXF86_SVGA' (to link the color server) or `make loadXF86_VGA16' (to link the 16 color server) or `make loadXF86_Mono' (to link the mono server).

8. Debugging

Debugging a new driver can be a painful experience, unfortunately. It is likely that incorrect programming of the SVGA chipset can lock up your machine. More likely, however, is that the display will be lost, potentially requiring a reboot to correct. It is HIGHLY recommended that the server be run from an attached terminal or a network login. This is the only rational way in which a debugger can be used on the server. Attempting to use multiple VTs for debugging is basically a waste of time.

Because of the potential for locking up the machine, it is a VERY good idea to remember to do a `sync' or two before starting the server. In addition, any unnecessary filesystems should be unmounted while the debugging session is going on (to avoid having to run unnecessary fsck's).

By default the server is built without debugging symbols. The server can grow VERY large with debugging enabled. It is very simple to rebuild your driver for debugging, though. Do the following:

1. Go to the driver directory.
2. Edit the Makefile. Look for the SECOND definition of `CDEBUGFLAGS'. Change this definition to

   ```
   CDEBUGFLAGS = -g -DNO_INLINE
   ```

   (this will enable debugging symbols and disable inlining of functions, which can make single-stepping a nightmare).
3. Remove the `sdc_driver.o' file.
4. Now follow the steps above for rebuilding the server. (Alternatively, instead of editing the Makefile, you can simply do `make CDEBUGFLAGS="-g -DNO_INLINE"' after removing the old .o file, then rebuild the server as described above).

This will give you a server with which you can set breakpoints in the driver functions and single-step them. If you are working in the source tree, and just learning about SVGA programming, it may be useful to rebuild vgaHW.c with debugging as well.

9. Advice

I cannot stress this enough - study all available references, and the existing code, until you understand what is happening. Do this BEFORE you begin writing a driver. This will save you a massive amount of headache. Try to find a driver for a chipset that is similar to yours, if possible. Use this as an example, and perhaps derive your driver from it.

Do not let the gloom-and-doom in the debugging section discourage you. While you will probably have problems initially (I still do), careful, deliberate debugging steps can bear fruit very quickly. It is likely that, given a good understanding of the chipset, a driver can be written and debugged in a day or two. For someone just learning about this kind of programming, a week is more reasonable.

10. Advanced Topics

Newer chipsets are getting into two advanced areas: programmable clock generators, and accelerated capabilities (BitBlt, line drawing, HW cursor). These are new areas, and the formal interfaces to them are not yet defined. It is advised that you contact the XFree86 team and get involved with the development/beta-testing team if you need to be working in these areas.

11. References

- Programmer's Guide to the EGA and VGA Cards, 3rd ed.
  Richard Ferraro
  Addison-Wesley, 1994
  ISBN 0-201-62490-7
  (This is the bible of SVGA programming - it has a few errors, so watch out. The third edition also covers several accelerated video cards.)
- vgadoc4.zip
  Finn Thoegersen
  (This is a collection of SVGA and other chipset documentation. It is available on most MS-DOS/Windows related FTP archives, including wuarchive. It is DOS/BIOS oriented, but is still extremely useful, especially for developing probe functions.)

12. Vendor Contact Information

ATI Technologies (VGA-Wonder, Mach8, Mach32, Mach64) 33 Commerce Valley Drive East

Thornhill, Ontario
Canada L3T 7N6
(905) 882-2600 (sales)
(905) 882-2626 (tech support)
(905) 764-9404 (BBS)
(905) 882-0546 (fax)

Chips & Technologies

???

Cirrus Logic (SVGA, Accelerators - CL-GD5426)

3100 West Warren Ave.
Fremont, CA 94538
(510) 623-8300 (N. CA, USA)
(49) 8152-40084 (Germany)
(44) 0727-872424 (UK)

Genoa Systems (GVGA)

75 E. Trimble Road
San Jose, CA 95131
(408) 432-9090 (sales)
(408) 432-8324 (tech support)

Headland Technologies, Inc (Video-7 VGA 1024i, VRAM II)

46221 Landing Parkway
Fremont, CA 94538
(415) 623-7857

Oak Technology, Inc (OTI-067,OTI-077)

139 Kifer Ct.
Sunnyvale, CA 94086
(408) 737-0888
(408) 737-3838 (fax)

S3 (911, 924, 801/805, 928, 864, 868, 964, 968, 764, 765)

(408) 980-5400

Trident Microsystems Inc (8800, 8900, 9000)

205 Ravendale Dr
Mountainside, CA 94043
(415) 691-9211

Tseng Labs Inc,

6 Terry Drive
Newtown, PA 18940
(215) 968-0502

Weitek (Power9000, 5186)

1060 E. Arques Ave,
Sunnyvale, CA 94086
(408) 738-5765

Western Digital

(714) 932-4900

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/VGADriv.sgml,v 3.13.2.1 1998/02/01
16:04:54 robin Exp $
```

```
$XConsortium: VGADriv.sgml /main/9 1996/10/28 05:13:22 kaleb $
```

Information for using/developing extern clock setting programs

The XFree86 Project, Inc.

16 December 1994

1. Using an external clock setting program

2. Developing an extern clock setting program

1. Using an external clock setting program

XFree86 provides a facility for setting the clock frequency on a graphics card by an external program. This is provided to make it possible to deal with cards that use clock selection methods not supported by the standard drivers.

This facility is enabled by adding a `ClockProg` line to the `Device` section of the `XF86Config` file. The format of this line is:

```
ClockProg "commandpath"
```

where *commandpath* is the full pathname of the clock setting program. No flags are allowed in *commandpath*.

When using this option, and no `Clocks` line is specified, it is assumed that the card has clocks which are fully programmable (like the SS24). However if the card has a fixed set of preset clocks a `Clocks` line is required in the Device section of the `XF86Config` file to tell the server which clock frequencies are available to it. The ordering of the clocks in the `Clocks` line should correspond to what the card/program expects. Up to 128 clock values may be specified.

The server calls the external program when it needs to change the clock frequency. This occurs at startup and when switching modes with the hot-key sequences. The command is passed two command-line arguments. The first is the clock frequency in MHz (as a floating point number -- currently specified to the nearest 0.1 MHz). The second argument is the index of the clock value in the Clocks list (the first clock is index 0). Cards with a fixed set of clocks would probably make use of the index, while cards with a fully programmable clock would use the frequency argument.

2. Developing an extern clock setting program

When such an external program is being used, the server does not change any register fields related to clock selection, and the external program must be careful to only modify clock selection fields. The program is run with stdin and stdout set to xf86Info.consoleFd -- which is the fd to use for display-related ioctl() operations if required. Stderr is the same as the server's stderr -- so error or warning messages should be directed there. The program is run with the uid set to the real user's ID -- so if it needs to use privileged system calls it should be suid-root. The program does not inherit any I/O access privileges, so it will need to do whatever is required to enable them.

The program is expected to return an exit status 0 when successful, and a status in the range 1-254 when it fails. If the external program fails during the server initialisation stage, the server exits. If it fails for a mode switch, the mode switch is aborted (the server assumes that the clock frequency hasn't been changed) and the server keeps running. If necessary an exit status may be specified in the future for which the server would exit if the program fails for a mode switch.

A sample clock switching program is provided for use with many ET4000 cards that have a fixed set of 8 clocks. This program is only intended as a sample, and it is not intended for general use (the internal server code handles this type of card). The program is `xc/programs/Xserver/hw/xfree86/etc/et4000clock.c` in the source tree, and `/usr/X11R6/lib/X11/etc/et4000clock.c` in the binary tree.

The idea of using an external clock program was suggested by Frank Klemm *<pfk@rz.uni-jena.de>*

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/clkprog.sgml,v 3.7 1997/01/24
09:32:48 dawes Exp $
```

```
$XConsortium: clkprog.sgml /main/3 1996/02/21 17:46:48 kaleb $
```

Information for Alliance Promotion chipset users

TOC

Henrik Harmsen (Henrik.Harmsen@erv.ericsson.se)

23 February 1998

1. Support chipsets
2. Acceleration
3. Configuration

TOC

1. Support chipsets

The apm driver in the SVGA server is for Alliance Promotion (www.alsc.com) graphics chipsets. The following chipsets are supported:

- 6422 Old chipset without color expansion hardware (text accel).
- AT24 As found in Diamond Stealth Video 2500. Quite similar to AT3D.
- AT25, AT3D AT3D is found in Hercules Stingray 128/3D. Most other Voodoo Rush based cards use the AT25 which is identical except it doesn't have the 3D stuff in it.

2. Acceleration

The apm driver uses the XAA (XFree86 Acceleration Architecture) in the SVGA server. It has support for the following acceleration:

- Bitblts (rectangle copy operation)
- Lines (solid, single pixel)
- Filled rectangles
- CPU->Screen colour expansion (text accel). Not for 6422.
- Hardware cursor

All in 8, 16 and 32 bpp modes. No 24bpp mode is supported. Also VESA DPMS power save mode is fully supported with "standby", "suspend" and "off" modes (set with with the "xset dpms" command).

3. Configuration

First: Please run the XF86Setup program to create a correct configuration.

You can turn off hardware cursor by inserting the following line in the Device section of the XF86Config file:

Option "sw_cursor"

Or turn off hardware acceleration:

Option "noaccel"

Please don't specify the amount of video RAM you have or which chipset you have in the config file, let the driver probe for this. Also please don't put any "clocks" line in the device section since these chips have a fully programmable clock that can take (almost) any modeline you throw at it. It might fail at some specific clock values but you should just try a slightly different clock and it should work.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/apm.sgml,v 1.1.2.2 1998/02/25
12:20:30 dawes Exp $
```

Information for Cyrix Chipset Users

The XFree86 Project Inc.

22 June 1999

1. Supported hardware

2. Features

3. XF86Config Option

4. Bugs and Limitations

5. Authors

1. Supported hardware

This driver (as used in the SVGA (VGA256), VGA16 and VGA_Mono servers) supports a single chipset `mediagx' that should work on the following Cyrix CPUs with integrated graphics:

- MediaGX
- MediaGXi
- MediaGXm

2. Features

- accelerated
- hardware cursor
- support color depths 1, 4, 8 and 16

Information for Cyrix Chipset Users : *Features*
Previous: *Supported hardware*
Next: *XF86Config Option*

3. XF86Config Option

Option "sw_cursor"

disable the hardware cursor.

Option "no_accel"

completely disables acceleration. Usually not recommended.

4. Bugs and Limitations

- On some older chipsets, the driver may trigger an illegal instruction just after probing for the ``scratchpad size''. If this is the case, email to hecker@cat.dfrc.nasa.gov with the output of

  ```
  XF86_SVGA -probeonly -verbose
  ```

  and this will be fixed.
- There are limitations to the modeline values that can be specified. Particularly, the difference between the first two horizontal timings (e.g. 640 656, 1024 1048) must be at least 16 and at most 24. The modeline values are not used in the 3.3.4 server since there is a static array used to load the registers. The modeline only identifies that a particular resolution is desired. The standard VESA modes up to 1280x768 are supported. For more specific information, consult the source code.
- The 4 colour server is slow due to the VGA banking mode used. Moreover, it does not work the way it is run by XF86Setup, which is probably due to the timing limitations.
- The 3.3.4 server MAY totally hang the machine at times. It is reported to be stable on a BSD platform using twm. It has crashed when using resolutions greater than 800x600 on a Linux (Debian based) system using wm as the window manager. The safest course is to use the 3.3.3.1 server instead. Efforts are under way to resolve this issue and provide a more robust server under the 4.x release.

5. Authors

- Annius Groenink <*Annius.Groenink@cwi.nl*>
- Dirk Hohndel <*hohndel@XFree86.org*>
- Brian Falardeau
- Special thanks to Cyrix and Wyse for helping us with the development of this server. Brian, a Cyrix employee, made the 3.3.4 update possible since the new 4.0 server has been our top priority.

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/cyrix.sgml,v 1.1.2.4 1999/06/24
06:16:44 hohndel Exp $
```

The Linux/m68k Frame Buffer Device

TOC

Geert Uytterhoeven (Geert.Uytterhoeven@cs.kuleuven.ac.be)

7 November 1998

1. Introduction
2. User's View of /dev/fb*
3. Programmer's View of /dev/fb*
4. Frame Buffer Resolution Maintenance
5. The X Server
6. Video Mode Timings
7. Converting XFree86 timing values into frame buffer device timings
8. References
9. Downloading
10. Credits

1. Introduction

The frame buffer device provides an abstraction for the graphics hardware. It represents the frame buffer of some video hardware and allows application software to access the graphics hardware through a well-defined interface, so the software doesn't need to know anything about the low-level (hardware register) stuff.

The device is accessed through special device nodes, usually located in the /dev directory, i.e. `/dev/fb*`.

2. User's View of `/dev/fb*`

From the user's point of view, the frame buffer device looks just like any other device in `/dev`. It's a character device using major 29, the minor specifies the frame buffer number.

By convention, the following device nodes are used (numbers indicate the device minor numbers):

0 = `/dev/fb0`

First frame buffer

32 = `/dev/fb1`

Second frame buffer

...

224 = `/dev/fb7`

8th frame buffer

For backwards compatibility, you may want to create a symbolic link from `/dev/fb0current` to `fb0`.

The frame buffer devices are also `normal' memory devices, this means, you can read and write their contents. You can, for example, make a screen snapshot by

```
cp /dev/fb0 myfile
```

There also can be more than one frame buffer at a time, e.g. if you have a graphics card in addition to the built-in hardware. The corresponding frame buffer devices (`/dev/fb0` and `/dev/fb1` etc.) work independently.

Application software that uses the frame buffer device (e.g. the X server) will use `/dev/fb0` by default (older software uses `/dev/fb0current`). You can specify an alternative frame buffer device by setting the environment variable `$FRAMEBUFFER` to the path name of a frame buffer device, e.g. (for sh/bash users):

```
export FRAMEBUFFER=/dev/fb1
```

or (for csh users):

```
setenv FRAMEBUFFER /dev/fb1
```


TOC

After this the X server will use the second frame buffer.

3. Programmer's View of `/dev/fb*`

As you already know, a frame buffer device is a memory device like `/dev/mem` and it has the same features. You can read it, write it, seek to some location in it and `mmap()` it (the main usage). The difference is just that the memory that appears in the special file is not the whole memory, but the frame buffer of some video hardware.

`/dev/fb*` also allows several ioctls on it, by which lots of information about the hardware can be queried and set. The color map handling works via ioctls, too. Look into `<linux/fb.h>` for more information on what ioctls exist and on which data structures they work. Here's just a brief overview:

- You can request unchangeable information about the hardware, like name, organization of the screen memory (planes, packed pixels, ...) and address and length of the screen memory.
- You can request and change variable information about the hardware, like visible and virtual geometry, depth, color map format, timing, and so on. If you try to change that informations, the driver maybe will round up some values to meet the hardware's capabilities (or return `EINVAL` if that isn't possible).
- You can get and set parts of the color map. Communication is done with 16 bit per color part (red, green, blue, transparency) to support all existing hardware. The driver does all the computations needed to bring it into the hardware (round it down to less bits, maybe throw away transparency).

All this hardware abstraction makes the implementation of application programs easier and more portable. E.g. the X server works completely on /dev/fb* and thus doesn't need to know, for example, how the color registers of the concrete hardware are organized. *XF68_FBDev* is a general X server for bitmapped, unaccelerated video hardware. The only thing that has to be built into application programs is the screen organization (bitplanes or chunky pixels etc.), because it works on the frame buffer image data directly.

For the future it is planned that frame buffer drivers for graphics cards and the like can be implemented as kernel modules that are loaded at runtime. Such a driver just has to call register_framebuffer() and supply some functions. Writing and distributing such drivers independently from the kernel will save much trouble...

TOC

4. Frame Buffer Resolution Maintenance

Frame buffer resolutions are maintained using the utility *fbset*. It can change the video mode properties of a frame buffer device. Its main usage is to change the current video mode, e.g. during boot up in one of your `/etc/rc.*` or `/etc/init.d/*` files.

Fbset uses a video mode database stored in a configuration file, so you can easily add your own modes and refer to them with a simple identifier.

5. The X Server

The X server (XF68_FBDev) is the most notable application program for the frame buffer device. Starting with XFree86 release 3.2, the X server is part of XFree86 and has 2 modes:

- If the `Display` subsection for the `fbdev` driver in the `/etc/XF86Config` file contains a

  ```
  Modes "default"
  ```

  line, the X server will use the scheme discussed above, i.e. it will start up in the resolution determined by `/dev/fb0` (or `$FRAMEBUFFER`, if set). You still have to specify the color depth (using the `Depth` keyword) and virtual resolution (using the `Virtual` keyword) though. This is the default for the configuration file supplied with XFree86. It's the most simple configuration, but it has some limitations.
- Therefore it's also possible to specify resolutions in the `/etc/XF86Config` file. This allows for on-the-fly resolution switching while retaining the same virtual desktop size. The frame buffer device that's used is still `/dev/fb0` (or `$FRAMEBUFFER`), but the available resolutions are defined by `/etc/XF86Config` now. The disadvantage is that you have to specify the timings in a different format (but `fbset -x` may help).

To tune a video mode, you can use fbset or xvidtune. Note that xvidtune doesn't work 100% with XF68_FBDev: the reported clock values are always incorrect.

6. Video Mode Timings

A monitor draws an image on the screen by using an electron beam (3 electron beams for color models, 1 electron beam for monochrome monitors). The front of the screen is covered by a pattern of colored phosphors (pixels). If a phosphor is hit by an electron, it emits a photon and thus becomes visible.

The electron beam draws horizontal lines (scanlines) from left to right, and from the top to the bottom of the screen. By modifying the intensity of the electron beam, pixels with various colors and intensities can be shown.

After each scanline the electron beam has to move back to the left side of the screen and to the next line: this is called the horizontal retrace. After the whole screen (frame) was painted, the beam moves back to the upper left corner: this is called the vertical retrace. During both the horizontal and vertical retrace, the electron beam is turned off (blanked).

The speed at which the electron beam paints the pixels is determined by the dotclock in the graphics board. For a dotclock of e.g. 28.37516 MHz (millions of cycles per second), each pixel is 35242 ps (picoseconds) long:

1/(28.37516E6 Hz) = 35.242E-9 s

If the screen resolution is 640x480, it will take

```
640*35.242E-9 s = 22.555E-6 s
```

to paint the 640 (xres) pixels on one scanline. But the horizontal retrace also takes time (e.g. 272 `pixels'), so a full scanline takes

```
(640+272)*35.242E-9 s = 32.141E-6 s
```

We'll say that the horizontal scanrate is about 31 kHz:

```
1/(32.141E-6 s) = 31.113E3 Hz
```

A full screen counts 480 (yres) lines, but we have to consider the vertical retrace too (e.g. 49 `pixels'). So a full screen will take

```
(480+49)*32.141E-6 s = 17.002E-3 s
```

The vertical scanrate is about 59 Hz:

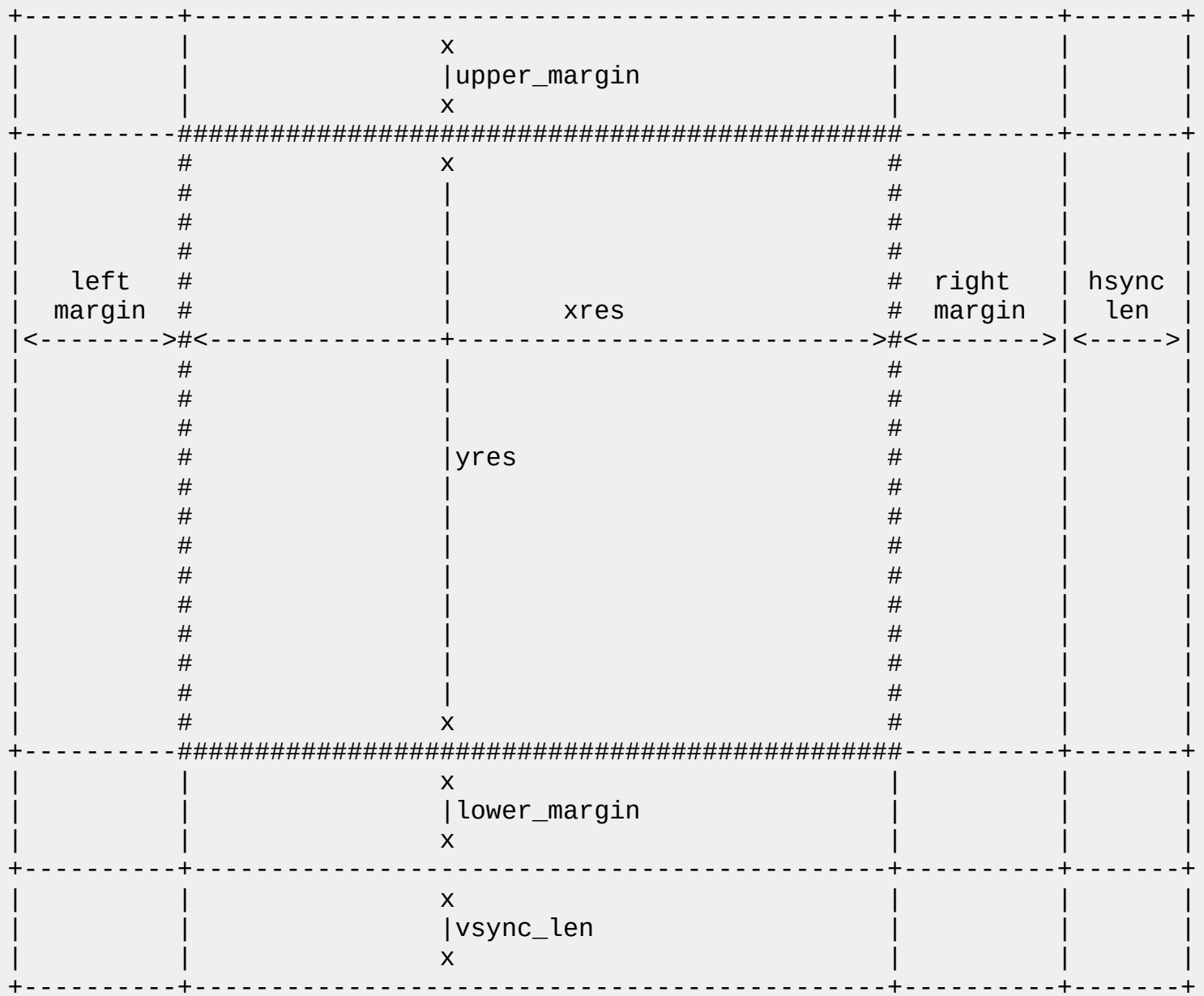
```
1/(17.002E-3 s) = 58.815 Hz
```

This means the screen data is refreshed about 59 times per second. To have a stable picture without visible flicker, VESA recommends a vertical scanrate of at least 72 Hz. But the perceived flicker is very human dependent: some people can use 50 Hz without any trouble, while I'll notice if it's less than 80 Hz.

Since the monitor doesn't know when a new scanline starts, the graphics board will supply a synchronization pulse

(horizontal sync or hsync) for each scanline. Similarly it supplies a synchronization pulse (vertical sync or vsync) for e new frame. The position of the image on the screen is influenced by the moments at which the synchronization pulses

The following picture summarizes all timings. The horizontal retrace time is the sum of the left margin, the right margin and the hsync length, while the vertical retrace time is the sum of the upper margin, the lower margin and the vsync length.



The frame buffer device expects all horizontal timings in number of dotclocks (in picoseconds, 1E-12 s), and vertical timings in number of scanlines.

7. Converting XFree86 timing values into frame buffer device timings

An XFree86 mode line consists of the following fields:

```
        "800x600"     50       800  856  976 1040    600  637  643  666
        < name >     DCF        HR  SH1  SH2  HFL     VR  SV1  SV2  VFL
```

The frame buffer device uses the following fields:

pixclock

pixel clock in ps (pico seconds)

left_margin

time from sync to picture

right_margin

time from picture to sync

upper_margin

time from sync to picture

lower_margin

time from picture to sync

hsync_len

length of horizontal sync

vsync_len

length of vertical sync

Pixelclock

- xfree: in MHz
- fb: In Picoseconds (ps)
- pixclock = 1000000 / DCF

Horizontal timings

- left_margin = HFL - SH2


TOC

- right_margin = SH1 - HR
- hsync_len = SH2 - SH1

Vertical timings

- upper_margin = VFL - SV2
- lower_margin = SV1 - VR
- vsync_len = SV2 - SV1

Good examples for VESA timings can be found in the XFree86 source tree, under `xc/programs/Xserver/hw/xfree86/doc/modeDB.txt`.

The Linux/m68k Frame Buffer Device **:** *Converting XFree86 timing values into frame buffer device timings*
Previous: *Video Mode Timings*
Next: *References*

8. References

For more specific information about the frame buffer device and its applications, please refer to the following documentation:

- The manual pages for fbset: fbset(8), fb.modes(5)
- The manual pages for XFree86: XF68_FBDev(1), XF86Config(4/5)
- The mighty kernel sources:
  - linux/drivers/video/
  - linux/include/linux/fb.h
  - linux/include/video/

9. Downloading

All necessary files can be found at

```
ftp://ftp.uni-erlangen.de/pub/Linux/LOCAL/680x0/
```

and on its mirrors.

10. Credits

This readme was written by Geert Uytterhoeven, partly based on the original `X-framebuffer.README` by Roman Hodek and Martin Schaller. Section `Converting XFree86 timing values into frame buffer device timings' was provided by Frank Neumann.

The frame buffer device abstraction was designed by Martin Schaller.

```
$XFree86: xc/programs/Xserver/hw/xfree68/doc/sgml/fbdev.sgml,v 1.1.2.6 1998/11/08
09:06:32 dawes Exp $
```

Information for Number Nine I128 Users

The XFree86 Project Inc.

24 October 1998

1. Supported hardware

2. Features:

3. Configuration:

4. Mode lines for the SiliconGraphics flat panel display:

5. Author(s)

1. Supported hardware

The current accelerated I128 server supports

- Imagine 128 (I128 with Texas Instruments TVP3025 or IBM528 RAMDAC). It has been tested with with 4MB of VRAM.
- Imagine 128 Ticket 2 Ride (I128-T2R with IBM526 or 528 RAMDAC). It has been tested with 4 MB and 8 MB of VRAM and DRAM.
- Imagine 128 Revolution 3D (I128-R3D with IBM526 RAMDAC). It has been tested with 4 MB, 8 MB, and 16 MB of WRAM or SGRAM.
- Imagine 128 Revolution IV (I128-R4 with SILVERHAMMER RAMDAC). It has been tested with 32 MB.

2. Features:

- uses linear frame buffer
- Resolutions up to the maximum supported by the card should be possible.
- 8 bpp, 16 bpp (depth 15 and 16), and 32 bpp (depth 24, sparse) are supported.
- supports RGB Sync-on-Green
- Makes use of the graphics accelerator.

3. Configuration:

The I128 driver should auto-detect all supported hardware so you needn't have anything other than the Identifier in the Section "Device" of the XF86Config file. When running the XF86Setup or xf86config programs one merely needs to select an I128 card so that the correct server will be used. One need not and should not specify a RAMDAC, clockchip or allow the setup program to probe for clocks. The driver will auto-detect the amount of video ram present.

The following Section "Device" options are supported by the MGA driver:

- Option "dac_8_bit"

  Will enable 8-bit DAC support.
- Option "no_accel"

  Will disable all hardware acceleration.
- Option "sync_on_green"

  Will enable syncing on green for sync-on-green monitors (these are typically fixed frequency workstation monitors).

4. Mode lines for the SiliconGraphics flat panel display:

- These mode lines are required for use with the T2R4 (Rev 4) and the SiliconGraphics Flat Panel display.
- Modeline "1600x1024d32" 103.125 1600 1600 1656 1664 1024 1024 1029 1030 HSkew 7 +Hsync +Vsync
- Modeline "1600x1024d16" 103.125 1600 1600 1656 1664 1024 1024 1029 1030 HSkew 5 +Hsync +Vsync
- Modeline "1600x1024d08" 103.125 1600 1600 1656 1664 1024 1024 1029 1030 HSkew 1 +Hsync +Vsync
- Modeline "800x512d32" 54.375 800 800 840 848 512 512 514 515 HSkew 7 DoubleScan +Hsync +Vsync
- Modeline "800x512d16" 54.375 800 800 840 848 512 512 514 515 HSkew 5 DoubleScan +Hsync +Vsync
- Modeline "800x512d08" 54.375 800 800 840 848 512 512 514 515 HSkew 1 DoubleScan +Hsync +Vsync

5. Author(s)

Robin Cutshaw, *robin@XFree86.Org*

and special help from:

- Galen Brooks, *galen@nine.com*

```
$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/I128.sgml,v 1.1.2.3 1999/01/02
02:32:16 robin Exp $
```