i810 with XFree86 4.x HOWTO

Toby Russell

May 2001

Revision History

Revision 1.1	2001−05−09	Revised by: tr
Revision 1.0	2001−05−01	Revised by: tr
Initial release.		

This HOWTO describes getting XFree86 4.x running on Intel's i810 graphics chipset by using special features of the 2.2.18 and 2.4.x kernels.

Table of Contents

1. Introduction

This document has a very specific purpose; to help people who are failing to get X working on Intel's i810 graphics chipset (hereafter "the i810"). It is written by a beginner (me), and it is imagined that it will be of use primarily to other beginners. The author would be flattered to hear that he has helped anyone more skilled than he. Furthermore, I know that the i810 works with XFree86 3.3.6, but I personally have not trod that path. My experience comes purely from XFree86 4.0 (hereafter "X4.x") and the i810/agpgart support available in the 2.2.18 and 2.4.x kernels, and consequently this HOWTO tackles that solution, or procedure, alone. The instructions that follow were written to the 2.4.x compile tune, but the procedure is similar enough to be translatable to the 2.2.18. Use your head, as Tony Buzan would say, and read the READMEs to be sure of any required alterations of method.

Even though I know this procedure works I feel obliged to point out that what I have recorded here is mostly that which I have worked out in my own bumbling way. It may well be that others know a quicker and more efficient method than that which follows. If so I will be happy to hear from them. As I mentioned previously, the i810 will work with XFree86 3.3.6, if one also uses some drivers designed by Intel for the task (namely XFCom_i810−1.2−3 and I810Gtt−0.2−4) but, in the interests of Linux purity, and of course knowing one does not have to use Intel's software, I recommend the method detailed here. It does not need Intel drivers.

Finally, no introduction would be complete without the following words of caution; this HOWTO should be regarded as a 'bare bones' set of instructions and should therefore be followed with all relevant README literature to hand. What follows is not exhaustive by any stretch of the imagination, and needs, at least for beginners, said README stuff.

2. Down to business

> **Note:** Do everything that follows logged on as root.

There are three distinct stages that need *not* be followed in the order listed here (please feel free to use your imagination). Said stages are;

- get and install X4.x
- get and compile kernel 2.2.18 or 2.4.x (including mknod agpgart stuff)
- nimbly tweak XF86Config

2.1. Getting and installing X4.x

The first stage is of course listed only as a guide for those who have perhaps tried getting XFree86 3.3.6 working with the i810 and failed, or perhaps those who have not even heard that X4.x supports the i810 and have been struggling vainly with their `XF86Config` file. I suppose the majority of people who find these instructions useful will have already loaded X4.x. You lot can skip this bit. Anyway, if you do need to know, X4.x can be got from; ftp://ftp.xfree86.org/pub/XFree86/4.0/binaries

But before you rush ahead and download away you must first be sure which version of X4.x suits your system. So download `Xinstall.sh` on its own and run (from within the folder containing `Xinstall.sh`):

`sh Xinstall.sh -check`

The results will direct you to the correct folder within the above mentioned URL from where the appropriate files for your system can be downloaded.

For a basic installation and to save time downloading one needs only the following absolute necessities, without exception (the others are optional and when included in the install process, I feel, increase the chances of things going wrong for the unwary and inexperienced):

`extract[.exe]`	`Xdoc.tgz`	`Xvar.tgz`
`Xbin.tgz`	`Xfenc.tgz`	`Xxserv.tgz`
`Xlib.tgz`	`Xfnts.tgz`	`Xmod.tgz`
`Xman.tgz`	`Xetc.tgz`	

Now knowing which set of files are suited to your system you can go ahead and download whichever suits. Then install with the following command (from within the folder containing freshly downloaded files):

`sh Xinstall.sh`

If you have been good everything will proceed smoothly. You will be asked some questions which the README file can explain/answer better than I. If something doesn't work as expected I refer you to the far more detailed, aforementioned README file, which you should definitely peruse. As a newbie I always read the readme files before downloading, installing, compiling and even getting up from my seat to go to the toilette. You can never be too sure.

That is the end of this stage.

2.2. Get and compile kernel 2.2.18 or 2.4.x (including mknod agpgart stuff)

You can get either kernel from ftp://ftp.kernel.com. Of course, read everything called README while you are at it. (In the README literature that comes with the 2.4.x kernel, there is an important note about where to unpack the source. Make sure you read it.) Put the kernel source file in `/usr/src/kernels`, and then run the following compile sequence, which I learned from a linuxnewbie article (to which you should refer if my directions are not clear enough for you, however it is specific to 2.2.x kernels). It can be found at the following address; http://www.linuxnewbie.org/nfh/intel/compiling/kernel_update.html. Of course, the location of the still−packed kernel is not really relevant, it only matters that it is unpacked to an acceptable location. OK, now for the commands:

`tar −xzvf /usr/src/kernels/linux−2.4.x.tar.gz`

or if you downloaded the better compressed bz2 version

`bzcat /usr/src/kernels/linux−2.4.x.tar.bz2 | tar xv`

and watch the screen spew out pages of information about what's happening. When it is finished it will have created a new `linux` folder.

OK, so, change to the new directory:

`cd linux`

and begin the compile process proper...

`make config`

Or preferably

`make menuconfig`

There's also **make xconfig**, but you haven't got X running, or you wouldn't be reading this. So that won't work. And I'm embarrassed to mention it in such an imperfect fashion but there is also something like **make oldconfig** but I can't find any reference to it in my books. In any case I am not addressing it here, though I am sure the procedure for it is very similar to that which follows for **make menuconfig**, should you be awkward and want to use it.

Now, I have gone through three text based kernel compiles (**make config**) and know how long winded they are. I reommend **make menuconfig** instead, which requires only that ncurses be loaded (you don't need X) and you will be taken through the pretty face of kernel recompilation. I loaded ncurses during a custom install of Red Hat 6.1, but I forget exactly at which stage that option is available. Otherwise ncurses is, I'm sure, on your distro's CD in rpm format, so if issuing **make menuconfig** just produces errors, install ncurses and try again.

The most relevant stages of the **make** process for solving our particular problem are:

- to select EXPERIMENTAL early on (by hitting return while the very first option is highlighted and then selecting the only suboption which is consequently revealed),
- towards the bottom of the base options, to enter "Character Devices" and select (not as "M" but as "*") "/dev/agpgart (AGP) support" (only available if the above instruction has been followed), and
- select the appropriate sub−option of "/dev/agpgart (AGP) support" (again not as a module "M" but as a static part of the kernel "*"), namely the "I810/I810 dc100I810e support" part.

  Note: The above explanation assumes you have run **make menuconfig** and so a little thinkology will be required to map it to a situation where **make** has been issued instead. But only a little.

(It has been pointed out to me that loading these features as modules would be more logical, since they are not required until **startx** is run. I have not tried the 'loadable module way' yet and will ammend this section of the HOWTO after I have tested it. I recommend the static mode here because I ran this procedure on a test version of the 2.4.x kernel and it was suggested to me that loading statically was a safer and stabler way to go. Now that 2.4.x is officially out there, perhaps modules will be more sensible. I'll let you know how it goes. (Thanks to Heron Ordonez for this.))

When all is over and you feel calm enough, do this;

`make dep`
`make clean` (not violently necessary but does no harm)
`make bzImage` (takes a while, this bit)
`make modules`
`make modules_install`

Now have a look at the `/boot` directory. You will probably see that `System.map` is a symbolic link to `System.map-[your_kernel_version]` and `vmlinuz` is a symbolic link to `vmlinuz-[your_kernel_version]`. This arrangement is true for many distros, but not all. I think some store `vmlinuz` in `/`, while `System.map` resides in `/boot`. Whatever the case is, use your brain and apply these instructions accordingly. So, basically you need to remove the symbolic links like so:

```
rm System.map
rm vmlinuz
```

Then new symbolic links need to be created to the about−to−be−copied−over−while−simultaneously−being−renamed, recently created files. It goes like this (assuming you have an i386 computer):

```
cp /usr/src/kernels/linux/arch/i386/boot/bzImage  /boot/vmlinuz-2.4.x
ln -s /boot/vmlinuz-2.4.x  /boot/vmlinuz
cp /usr/src/kernels/linux/System.map  /boot/System.map-2.4.x
ln -s /boot/System.map-2.4.x  /boot/System.map
```

Tip: You don't need to use absolute pathnames if you are in `/boot`. But if you are the excessively cautious type and do use absolute pathnames, you just have longer names for

> your symbolic files. In fact the whole symbolic link thing here is only necessary if you want to play it that way. Essentially, minimalistically, you can have one kernel called `vmlinuz` and name all the others by their version number (or just trash them!), and swap all the names around when you want to boot another kernel. Or give each kernel a unique name, and have one entry per kernel in `/etc/lilo.conf`. It's up to you.

Now you need to edit `/etc/lilo.conf`. This is achieved thusly:

```
image=/boot/vmlinuz
label=[what-ever-you-want-that-is-relevant-easy-to-type-and-remember]
read-only
root=/dev/hda[n]
```

After editing `lilo.conf` you must do this:

```
/sbin/lilo
```

so that the crisp, shiny, new linux kernel be made known to lilo, otherwise (I have experienced this) the new kernel will not be available for booting. Which would be silly. So after all this take a deep breath and reboot, select your new kernel and with fingers crossed, watch. It should work. If it does, go and celebrate a little. But don't let it get to your head because you have yet to mknod the agpgart module, a simple yet essential procedure done thusly:

```
cd /dev
mknod agpgart c 10 175
```

which basically creates the very essential character device (X won't run without it) driver which acts kinda like a 'go–between' for the i810 chipset and the X server. (Thanks to Heron Ordonez for saving me some embarrassment here.) Pretty scientic stuff. Sorry about that.

That was the end of this stage.

2.3. Nimbly tweak XF86Config

I've done a lot of this and it get's mighty tedious when it fails 23 times in a row I CAN TELL YOU, so pay attention and read very closely the man page (run **`man XF86Config`** at the command prompt). First of all I recommend running the in–no–way–user–friendly **xf86config** (*observe case!*) to genertate a base `XF86Config` file as the other tools seem to produce `XF86Config` files which are in my experience incompatible with X4.x. When you run through the questions **xf86config** asks and you reach the card section, there will be nothing for you to choose, so choose that very nothing. You'll be entering the right stuff later, after the base file has been created. Then, after answering all the questions as well as you can, save the file as `/etc/X11/XF86Config`.

So, finally, the all important addition is:

```
Section "Device"
Identifier  "i810"
```

```
Driver  "i810"
VideoRam "4096"
```

and it should be inserted in the Graphics Device Section. There should in any case be an existing "Device" section which you could edit if you prefer. From thereon you should, having defined the i810 for X, enter "i810" wherever you see a "Device" field. I am including a couple of sections from my `XF86Config` file as an example, and hopefully to make a little clearer what I mean:

```
Section "Device"
Identifier  "i810"
Driver  "i810"
VideoRam "4096"

Section "Screen"
Identifier  "Screen 1"
Device  "i810"
Monitor "Highscreen  17inch"
DefaultColorDepth 24
SubSection "Display"
Depth  8
Modes  "1024x768"
EndSubSection
SubSection "Display"
Depth  15
Modes  "1024x768"
EndSubSection
SubSection "Display"
Depth  16
Modes  "1024x768"
EndSubSection
SubSection "Display"
Depth  24
Modes  "1024x768"
EndSubSection
SubSection "Display"
Depth  32
Modes  "1024x768"
EndSubSection
EndSection
```

Warning

As you can see I have only given X the option of "1024x768", and have a default colour depth of 24 bits, because I like it that way, and the i810 can easily cope with that resolution and depth on my 17" monitor. How that would work on a 21" I do not know. Experimentation will teach you.

I am going to be boring and say it again, but a more complete understanding than I can give here of the mysteries of the `XF86Config` file can be achieved by closely reading the man page (see above). This is really important if you want to have a chance of solving any problems that are bound to come up now and again, that have not been covered here.

That should do it. Now save `XF86Config` and run:

`startx`

It should work. It did for me. You will be happy. If not contact me at <trussl@hotmail.com> and I will endeavour to help you.

> **Note:** This is a kind of a p.s. to this section but may be helpful. I had a wee problem when going through the XF86Config part of this HOWTO during a test run. It stemmed from having read but not fully understood some blurb about the i810 and X4.x not working at all resolutions with a buffer extension (or something like that). Anyway, I made no notes about this and cannot therefore remember exactly what I read. Because I remember this vaguely I can only say the following with certainty; you need the following stanza at the beginning of your `XF86Config` file:

```
# This loads the DBE extension  module
Load "dbe"  # Double buffer  extension
# This loads the miscellaneous extensions module, and
disables
# initialisation of the XFree86-DGA extension within  that
module.
SubSection "extmod"
Option "omit xfree86-dga" # don't initialise the  DGA extension
EndSubSection
```

> So if X reports errors about a "shape extender" or "shape extension", you may well find that your `XF86Config` file is missing the above listed stanza.

3. Thank you

I must point out that I would not have known how to fix the i810 and X4.x problem if it were not for the pioneering efforts of Val Henson who guided me through the process and recommended the 2.4.x kernel in the first place. And now that this is an ammended version, I would also like to thank Heron Ordonez for pointing out a few problems which I have in part addressed, and will fully address in due course. Curtis Stone pointed out to me that the features I thought only available in the 2.4.x kernel were present in 2.2.18. Thanks to him too. I am now also endebted to Cameron Kerr for pointing out that the 2.4.x kernel must not be unpacked in /usr/src/linux. I had had no success with the 2.4.x until this was pointed out to me, but would have been OK had I read the accompanying README, ie followed my own instructions.

If this process carries on in this fashion the 'Thank you' will one day be the largest section of this HOWTO. This is an open process and all comments (politely phrased of course!) are welcome.