The Linux Ultra−DMA Mini−Howto

Table of Contents

The Linux Ultra−DMA Mini−Howto

Brion Vibber, `brion@pobox.com`

v3.01, 6 December 2001

This document is intended to explain how to use Ultra−DMA aka Ultra−ATA aka Ultra33 and Ultra66 hard drives and interfaces with Linux. The most recent version of this mini−Howto can be obtained in HTML format at http://pobox.com/~brion/linux/Ultra−DMA.html.

---

1. Introduction and Disclaimer

This document is intended to explain how to use Ultra−DMA aka Ultra−ATA aka Ultra33 and Ultra66 hard drives and interfaces with Linux. In many cases there is no difficulty in using them, but some tweaking can increase performance. In other cases, you need to go to extraordinary lengths simply to access your hard drives.

1.1 Disclaimer

The information in this is document is, to the best of my knowledge, correct, and should work. However, there may be typos, there may be mysterious transmission errors, and there may be strange incompatibilities within your own system that prevent the techniques described herein from working properly. So... before you go fiddling around with you hard drive, **BACK UP ANY DATA YOU WANT TO KEEP!** If you are not already performing regular backups, please start doing so for your own good.

1.2 Credits

Michel Aubry − UDMA−enabled VIA−related patch for <=2.0.33 & more info, grand unified UDMA patch for 2.0.34+

Andrew Balsa − Provided some general UDMA info and the udma−generic patch for Intel TX, SiS, and VP1 on <=2.0.33; also the grand unified UDMA patch for 2.0.34+

Maxime Baudin − French translation

Bokonon − ``Controller'' vs. ``interface''

John G. − VIA VP2 patch for <=2.0.33 & info

Martin Gaitan − Promise Ultra33 ide0/ide1 installation workaround

Andre M. Hedrick − current Linux IDE subsystem maintainer

Håvard Tautra Knutsen − Norwegian translation

Norman Jacobowitz − Bugged me to add info on the VP3

John Levon − Info on TX Pro mobos

Peter Monta − Info on using two Ultra33 cards

Masayoshi Nakano − Japanese translation

Gadi Oxman − The Promise Ultra33 patch for <=2.0.34 & finding the secret numbers for the workaround

Andy Pearce − Suggested adding info on the additional device files for hde−h

Andrei Pitis − LILO patch

Brion Vibber − The document itself

1.3 Document History

v3.01, 6 December 2001: Relicensed under GNU Free Documentation License; no content changes.

v3.0, 9 November 1999: Finally found time to update some key changes such as the relocation of the IDE patch archive to the Kernel.org archives... pesky school! Updated all sunsite links to new metalab.unc.edu or www.linuxdoc.org

v2.1, 27 May 1999: Corrects some minor omissions and errors from 2.0 and adds information on the Promise Ultra66 and 2.2/2.3 kernels.

v2.0, 7 August 1998: Major updates and almost total restructuring of the document into onboard (motherboard) and offboard (add−in cards) interfaces; the Grand Unified UDMA patch(a part of the Jumbo patch) for 2.0.35. Put credits in alphabetical order by last name. Changed ``controller'' to ``interface'' in many cases to be more technically correct. Added info on enabling/disabling UDMA, the blacklist, and more!

v1.45, 6 July 1998: Minor updates − Red Hat 5.1 and 2.0.34 patch for Promise Ultra33, LILO patch for booting off of PCI interfaces such as the Promise Ultra33

v1.41, 3 May 1998: Fixed a couple of typos, added translators to credits.

v1.4, 28 April 1998: UDMA−Generic patch, some more general info. Copying section added.

v1.3, 5 March 1998: VIA VP3 info, better patching instructions, pointer to more recent Promise patch.

v1.2, 27 January 1998: Additional Promise workaround info.

v1.1, 21 January 1998: New info about VIA chipset, installing *around* the Promise Ultra33, and enabling Bus Master & UDMA transfer modes.

v1.0, 19 January 1998: More or less complete, first version done in SGML.

1.4 Copying

2. What is Ultra−DMA and why do I want it?

Here's a brief overview of IDE−based drive technologies:

2.1 IDE, EIDE, & ATAPI

These are older drive technologies. Most non−SCSI hard drives and drive interfaces that you can buy today or are likely to be using are EIDE, although many of the larger drives now available are UDMA.

2.2 Bus Master DMA

Bus Master DMA is a technology for increasing the speed of hard disk data transfers which requires support from the motherboard and the BIOS, and at least some support from the drive.

You can learn more at http://developer.intel.com/design/pcisets/busmastr/FAQs.htm.

2.3 Ultra−DMA aka Ultra−ATA aka Ultra33 aka...

Ultra−DMA has many names, but we'll just call it UDMA in here.

UDMA is a more advanced technology which provides for even faster throughput, up to 33.3 MB/s in UDMA mode 2 and 66.7 MB/s in UDMA mode 4, twice to four times that of EIDE, for much lower prices than SCSI. Many new computers come with large UDMA drives and UDMA interfaces, and it's possible to add a UDMA interface card (such as the Promise Ultra33 or Ultra66) to an existing system to boost speed, even on older non−UDMA drives.

You can learn great details about UDMA at http://www.quantum.com/src/whitepapers/ultraata/

Note that cable length should be kept shorter for UDMA, compared to plain DMA, preferably less than 30 cm (12") maximum length though 18 inches will usually be fine. 66 MB/s requires a special 80−conductor cable and should definately not be longer. If you get a lot of CRC errors, try using a shorter cable.

2.4 Just how ``Ultra'' is it anyway?

Before we get any farther, let's clear up a misconception. That 33 or 66 MB/sec figure is the **burst transfer rate**, and it's not something you're going to see very often. To explain, here is a clip from udma-generic's UDMA.txt:

```
Burst (instantaneous) transfer rates are supposed to go from 16.6MB/s (PIO
mode 4) to 16.6MB/s (DMA mode 2) up to 33MB/s (UDMA). In his patch against
kernel 2.1.55, Kim-Hoe Pang actually checked the UDMA burst transfer rate
with a logic analiser: 60ns/word, which translates into 33MB/s.

Note that burst transfer rates only affect data transfers to/from the EIDE
drive cache (476kB for the IBM 6.4GB drive), and IMHO are not particularly
relevant for most Linux users.

The Linux kernel uses as much RAM as possible to cache hard disk data
accesses, and so if data is not in the kernel cache there is little chance
that it will be in the (much smaller) hard disk cache.
```

Much more relevant is the **sustained transfer rate**, the speed at which data can be transferred from the drive to main memory where it can be used. An easy way to measure the sustained transfer rate is to use `hdparm`, for instance ``hdparm -Tt /dev/hda'' to measure the rate of the first IDE device.

```
Here is some data gathered after extensive testing, using the hdparm utility
(also written by Mark Lord):

PIO mode 4 transfer rates under Linux:   +/- 5.2MB/s

DMA mode 2 transfer rates under Linux:   +/- 7.2MB/s

UDMA mode 2 transfer rates under Linux:  +/- 9.8MB/s
```

As you can see, UDMA is still almost twice as fast as plain EIDE and significantly faster than plain bus mastering DMA. Most current UDMA drives will give you between 10 and 15 MB/s using UDMA mode 2 (33 MB/s) or 4 (66 MB/s) enabled.

Also, using DMA vastly reduces CPU usage during disk I/O vs PIO.

2.5 How does UDMA compare to SCSI?

I don't have any hard numbers to give you, but the general consensus is that high−end SCSI can give better performance than UDMA. However if you've looked at the price tags on any hard drives lately you'll notice that UDMA drives tend to be much less expensive. The performance/price ratio favors UDMA in most cases.

---

3. Using your UDMA hard drive with an EIDE interface

This is easy to do. Since all UDMA drives are fully EIDE backward−compatible, just plunk your drive on your EIDE interface like it was any old hard drive and Linux should have no problems detecting or using it. However, you will of course be limited to the slower speed of EIDE.

4. Using your hard drives with a UDMA interface

Well, there is good news and there is bad news. The good news is that a UDMA interface can be used with both UDMA hard drives and legacy EIDE hard drives, and will be a lot faster than an EIDE interface.

The bad news is that the old stock kernels (2.0.x) do not currently support UDMA very well. The new 2.2.x kernels do support UDMA33, however, and kernel patches are available to add UDMA support for kernels that lack it.

In addition, certain UDMA interfaces that are add−in cards rather than built into the motherboard require either a patch or some trickery to use on older kernels. That is why this document exists − to explain how to get the patches and work the trickery.

5. Offboard PCI UDMA interfaces

These are UDMA interfaces on PCI cards that can be used to add UDMA support to an existing computer without replacing the motherboard, or for adding support for an additional four drives to a machine which has had its onboard interfaces filled. They can also be found preinstalled in some computers, especially Gateway 2000 and Dell machines.

Most of them are not supported by the old stable kernels (2.0.x), but many should work with a 2.2.x kernel − the Red Hat 6.0 and SuSE 6.1 distributions are based on 2.2.x kernels, as are the most recent versions of most other distros. However some of the latest cards (the Promise Ultra66 for instance) won't work even with the current 2.2.x kernels, if you have this or can't get a newer distribution then you must apply a kernel patch or upgrade to a newer kernel version. If you need to install Linux onto a hard drive on one of these interfaces in this case, you will need to use a few odd tricks.

5.1 Promise Ultra33

This is a PCI card that has two UDMA channels on it, supporting up to four drives total. You can look up specifications & pricing at http://www.promise.com. This card shipped in early model Gateway 2000 Pentium II systems.

Kernels 2.0.35 and later and all 2.2.x kernels support the Ultra33 and you should have no trouble installing a distribution that uses these kernels. However, the older stable kernels (2.0.34 and below) do not, and since most older Linux distributions include these older kernels it can be a little difficult to get Linux installed if you can't or don't want to use a newer version (for instance if you are standardized on a particular version of a distribution throughout your organization).

Installing Linux with the Ultra33

Although there is a patch for the Ultra33 interface, it is not very easy to apply a patch and recompile your kernel if you have not installed Linux yet! So, here is a workaround which allows you to install. Thanks to Gadi Oxman for the following information on getting the interface settings:

```
        If we can access the console with the installation disk, we can also
```

```
use "cat /proc/pci" to display the Promise interface settings:

    RAID bus interface: Promise Technology Unknown device (rev 1).
      Vendor id=105a. Device id=4d33.
      Medium devsel.  IRQ 12.  Master Capable.  Latency=32.
      I/O at 0xe000.    (a)
      I/O at 0xd804.    (b)
      I/O at 0xd400.    (c)
      I/O at 0xd004.    (d)
      I/O at 0xc800.    (e)

and pass "ide2=a,b+2 ide3=c,d+2" as a command line parameter to the kernel.
```

Note that the numbers probably are not the same as what you will have. Just as an example, the parameters to use for the above set of numbers would be ``ide2=0xe000,0xd806 ide3=0xd400,0xd006''. If you are only using the first channel on the Ultra33 (for instance, if you only have one drive, or two if they are master and slave on the same channel), then you won't need to specify `ide3`.

Red Hat 5.1: Boot with the boot diskette and press enter when prompted. The kernel will load, and then you will be asked for a language, keyboard type, and installation method. You may be prompted for additional information about the source media; it doesn't matter right now what you tell it as long as you can get to the next step. Next you should see a screen titled ``Select Installation Path''; press Alt−F2 now to get to a command prompt. Run ``cat /proc/pci'', write down the numbers as above, and reboot from the boot disk. This time, type ``linux ide2= *(this is where you put the numbers like shown above)* ide3=*(more numbers)*''. It should now be able to install onto your hard disk without difficulty, however LILO will probably not be able to install; instead make a boot floppy and boot it with the same parameters until you can patch LILO and and the kernel.

Red Hat 5.0 and **Slackware 3.4:** These are similar, but with the wrinkle that the setup programs ignore `/dev/hde−h` (the drives on `ide2` and `ide3`). In order to install to or from these drives it is necessary to override one or both of the onboard interface's channels. However be sure not to override a device that you need to install; for instance if you are installing from a CD−ROM drive on `/dev/hdd` (`ide1` − onboard interface) to a hard drive on `/dev/hde` (`ide2` − the Ultra33), you should override the non−essential `ide0` with `ide2` and leave `ide1` intact. Assuming the numbers above you would boot with ``ide0=0xe000,0xd806''. Red Hat 5.0 will give you a shell prompt if you use the rescue disk capability, and Slackware includes a shell in the regular installation process. However Red Hat 5.0 is difficult to boot after installation; if you have problems you could try downloading a Slackware boot disk from ftp://ftp.cdrom.com/pub/linux/slackware−3.5/bootdsks.144/ and using that to boot.

With another Linux distribution you will have to improvise a bit, but the process should be about the same as the above.

IMPORTANT: Without the patch (discussed in the section Unified IDE), the kernel **needs** these boot parameters in order to access your hard disk! Therefore it is very important that when you configure LILO, either on the hard disk or on a boot floppy, that you give it the **exact same parameters** that you gave when installing. Otherwise your system won't boot! It should be possible to give them to LILO when you boot (ie, press Shift, type in ``linux ide2=.....'' each time you boot), but only if you kept the numbers! It is recommended that you patch your kernel as soon as possible so you will not have to worry about that anymore; once you are booting with a patched kernel, you can get rid of the boot parameters. Also, as far as I know there is no way to pass boot parameters to a plain kernel boot floppy (as made with ``make zdisk''), you **must** use LILO or another loader (such as LOADLIN) that lets you pass boot parameters.

However, unpatched kernels and installation programs often have a difficult time actually using ide2 and ide3, even if the drives are detected properly. So if you can't get Linux to install using the above technique, try specifying ide0 or ide1 instead of ide2 or ide3 (thanks to Martin Gaitan for this technique). This essentially replaces the on−board interface with the Promise Ultra33 as far as the kernel is concerned, and you can follow the directions in the next section as if you had physically moved it. Note that if you're using an IDE CD−ROM drive connected to your on−board interface to install from, you will want to make sure that you do not take over the interface that the CD is on or you will not be able to install! If the CD is hda or hdb, use ide1 for your hard drive, and if it is hdc or hdd, then use ide0.

Installing Linux Around the Ultra33

If you cannot get the software workaround to work, you will have to try a more brute force approach. Here's an alternative method that is virtually guaranteed to work, but will require you to open up your computer and mess about in it. **NOTE:** If you are not familiar with the process of connecting and disconnecting IDE drives, **read the manuals** that came with your computer, your hard drive, and/or the Promise Ultra33 before attempting this! If you screw something up and don't know how to put it back, you could end up being sorry!

That being said, it's all really quite simple. Most motherboards these days have built−in EIDE interfaces. Disconnect your hard drive from the Ultra33 and connect it to the onboard interface. If you have other IDE devices, such as a CD−ROM, tape, or ZIP drive, on your oboard interface, it is easiest if you either add the hard drive on an unused channel (the secondary instead of the primary) or temporarily displace a device that you don not need immediately (such as ZIP or tape). Install Linux. Download and apply the Promise UDMA patch (see next section).

Now you are ready to move the drive back onto the Promise... almost. To be safe, make a kernel−image boot floppy (`cd /usr/src/linux ; make zdisk`), which you will be able to use to boot your system in case LILO doesn't work. Actually, to be *very* safe, make two and put one away for now.

Okay, now it is time to think a little... if you have just one hard drive and it is going to be on the Promise, then it will most likely be `/dev/hde` (`a` and `b` are for the primary onboard interface, `c` and `d` for the secondary onboard interface). If you are going to put any other drives on it, then the slave of the Promise's first channel will be `/dev/hdf`, the master of the second will be `/dev/hdg`, and the slave of the second will be `/dev/hdh`.

Edit `/etc/fstab`, and change all the partitions of the hard drives you are moving from the onboard drives (`/dev/hda`, `hdb`, etc) to their new locations on the Promise (`/dev/hde`, `hdf`, etc). If you had to displace any devices (such as a CD−ROM or ZIP drive) that you want to leave on the onboard interface, then change them to their new locations as well. For instance, if your CD−ROM was originally the master on the primary channel (`/dev/hda`), but you put your hard disk there and had to bump the CD to the slave (`/dev/hdb`) or to the secondary channel (`/dev/hdc`), and now you want to put it back, then change it to `/dev/hda`.

If you are using LILO, reconfigure LILO to use the new location of the drive (LILO configuration is beyond the scope of this document, if you do not know how, read the LILO mini−HOWTO), or else it probably will not be able to boot unless you use that boot floppy I had you make, which you will also want to configure to boot off the new partition. This is done using the `rdev` command. Put the floppy in the drive and type ``` ``rdev /dev/fd0 /dev/hde1'' ```. Of course that's assuming your root partition is the first on your first UDMA drive. If not (mine is `/dev/hde7`, for instance), then obviously use the appropriate partition number!

Reboot. Your system should now work fine.

Patching for the Ultra33

Kernels 2.0.35 and later support the Promise Ultra33 natively; download an upgrade from your Linux distribution or from http://www.kernel.org.

For instructions on how to compile the kernel, read the Kernel HOWTO.

Using two Ultra33 cards in one machine

This is currently not working correctly... don't do it right now unless you're willing to fiddle with the kernel to try to get things to work.

5.2 Promise Ultra66

This is essentially the same as the Ultra33 with support for the new UDMA mode 4 66 MB/sec transfer speed. Unfortunately it is not yet supported by 2.2.x kernels.

There is a patch for 2.0.x and 2.2.x kernels availabe at http://www.kernel.org/pub/linux/kernel/people/hedrick, and support is included in the 2.3.x development kernel series at least as of 2.3.3.

However to get far enough to patch or upgrade the kernel you'll have to pull the same dirty tricks as for the Ultra33 as in the section above, or else use a boot disk image provided by Promise

5.3 Artop ATP850UF

This card is supported by the unified IDE code. Installation of Linux onto a system with one of these as the interface for the target disk may be similar to the workarounds for the Promise Ultra33.

5.4 Adding device files

The tertiary and quaternary IDE interfaces (ide2 and ide3) use device files of the form `/dev/hde*` through `/dev/hdh*`. On older kernels these devices were not automatically created, so you may need to add them manually for things to work properly.

This can be done easily if you have a current copy of the Linux kernel source installed; simply run `/usr/src/linux/scripts/MAKEDEV.ide` and it will create all relevant device files.

---

6. Onboard UDMA interfaces

These are UDMA-capable drive interfaces built into motherboards. They use the standard IDE I/O ports and so are fully usable at the slower non-UDMA speeds on an unpatched 2.0.x kernel such as are used when installing Linux. Thus they should not cause any difficulties during installation, and patching for UDMA speed is a welcome luxury instead of a necessary step. Some UDMA support is in the latest 2.0.x kernels I believe, and is built into current 2.2.x kernels for the Intel chipsets.

6.1 Intel FX, HX, VX, TX, LX, and BX

Thanks again to Gadi for this info:

```
Bus mastering DMA support for the Intel TX chipset is available in 2.0.31
and above.
```

In older kernels (such as Slackware 3.4's 2.0.30), the interface will be used in the slower EIDE mode. In either case the interface will be automatically detected by the kernel and you should have no trouble using it.

Full UDMA mode 2 support for these chipsets is included in 2.2.x kernels and the unified IDE patch; see Unified IDE.

6.2 The VIA VP2 and Related Chipsets

This interface also can be autodetected and used in EIDE mode by an unpatched kernel, but if you have one of these, you will want to grab a patch so you can get faster throughput and do away with annoying "unkown PCI device" messages.

One is available at http://www.ipass.net/~prefect/; it is designed for the VIA VP2/97 chipset, found on FIC's PA−2007 and PA−2011 motherboards, but may work on related chipsets. It has been reported that it functions on the newer VIA VP3 chipset, your mileage may vary.

Note that this patch only supports Bus Mastering mode, not full UDMA mode, but it's still better than plain−vanilla EIDE mode. Follow the directions at the patch's site for enabling BMDMA mode.

There is another patch that supports full UDMA mode at http://www.pyreneesweb.com/Udma/udma.html, designed for the VIA VT82C586B, and it ought to work on the VP2, VP3, VPX, P6 and AGP Apollo chipsets. Follow the directions for installation and UDMA enabling there, but it is recommended that you back up any data you want to keep, as there are potential problems with incompatible motherboards. But, if it does work, it should work without problems.

Note that the VP1 chipset is not known to work with these patches, but is supported by the Unified IDE patch.

6.3 TX Pro and other ``Pro'' boards

UDMA is not currently supported for the TX Pro motherboards. They are not the same as a TX mobo, and apparently misreport their DMA capabilities hence the problem. Someone is working on this I hear, so a patch may appear some time in the future but not yet.

6.4 HPT 366

This chipset is on the popular Abit BP−6 motherboard and others, and provides UDMA mode 4 66MB/s support on two generals, generally in addition to two other mode 2 33MB/s channels. It is supported by the current unified IDE code but not in any current release kernels. Installation thus may require workarounds similar to the Promise Ultra33 did on older 2.0.x kernels.

7. Unified IDE Patches

The unified IDE patches provide support for many chipsets and offboard cards, and are available for 2.0.x, 2.2.x, and the 2.3.x development kernels. If your chipset isn't supported by a current stock kernel, you'll want to patch it with these.

The unified IDE code is maintained by Andre Hedrick, and patches are available at your local kernel archive mirror.

UDMA support is provided for at least the following chipsets, and probably many more I don't know about:

- All Intel chipsets: FX, HX, VX, TX, LX
- All SiS chipsets (only SiS5598 tested, but this entire family of chipsets has the same bult−in 5513 interface device).
- VIA chipsets (only 82C586B tested, but again this family of chipsets has the same interface structure). Special diagnostics support is available for the VIA interfaces.
- Promise and Artop PCI UDMA interface cards support.
- Aladdin V (ALi15x3) chipset
- HPT343 board and HPT366 onboard chipset (caveat, see Abit BP−6)

It is also designed to be easy to extend to support other chipsets.

Here are a few notes from Andre Balsa, the author of an earlier patch:

```
Performance with IBM UDMA drives on a good motherboard approches the
maximum head transfer rates: about 10 Mb/s (measured with hdparm −t −T).

The Intel TX chipset has a single FIFO for hard disk data shared by
its two IDE interfaces, so using 2 UDMA drives will not yield such a
great improvement over a single UDMA drive.
However, the SiS5598 has two completely separate interfaces, each with
its own FIFO. Theoretically, one could approach 66Mb/s burt transfer
rates on motherboards with the SiS5598 chip, using the md driver and
data striping over two drives. The SiS5571 has the same interface
architecture, I think. I don't have the VIA chipsets datasheets, so I
can't say anything about those.

The Linux IDE (U)DMA kernel driver by Mark Lord has a particularly
low setup time (i.e. latency for data transfers). It is ideal for
frequent, small data transfers (such as those in Linux news servers),
and might be in some cases superior to its SCSI counterparts.
```

8. Activating and Deactivating UDMA

Normally, a UDMA−aware kernel will automatically enable UDMA support for drives and interfaces that support it. In most cases that it doesn't, the kernel either doesn't know how to drive your IDE chipset (get yourself a patch, see above) or doesn't believe it is safe to enable it (meaning you shouldn't!).

However in some cases the drive is capable of UDMA but the BIOS drops the ball and doesn't report it properly, and forcing the issue can be useful.

8.1 Using kernel boot parameters

On kernels 2.1.113 and up, you can enable DMA for both drives on a given IDE interface using the `ideX=dma` kernel parameter, where X is the number of the interface (the first is 0). This may not actually force UDMA though.

Kernel boot parameters can be set using LILO, LOADLIN, or most Linux boot loaders. For more information see the Bootdisk HOWTO.

8.2 Using hdparm

`hdparm` is a program used to tweak the parameters of hard drives under Linux. Among other things you can use it to enable or disable UDMA for a drive and test its sustained transfer rate.

The current version of `hdparm`, is 3.6 as of this writing. Unpatched older versions will not properly report or set information on UDMA, so be sure to upgrade! You can obtain the source code for hdparm 3.6 at http://metalab.unc.edu/pub/Linux/system/hardware/hdparm−3.6.tar.gz.

Compile and install it something like this:

```
tar zxvf /tmp/download/hdparm-3.6.tar.gz
cd hdparm-3.5
make
su root
(type password when prompted)
make install
cp /usr/local/sbin/hdparm /sbin/hdparm
exit
```

To enable DMA for a hard drive: `hdparm −d1 /dev/hda`

To disable DMA for a hard drive: `hdparm −d0 /dev/hda`

To measure transfer rate of a hard drive: `hdparm −Tt /dev/hda`

To see what options are enabled for a hard drive: `hdparm /dev/hda`

To see more info on your drive than you wanted to know: (this will show which UDMA modes are supported/enabled) `hdparm −i /dev/hda`

For more detailed info (such as how to choose which UDMA mode to use) read the man page (``man 8 hdparm'').

---

9. Problems

9.1 The UDMA Blacklist

The following drives are ``blacklisted''. You should **not** use UDMA with these drives as it may cause corruption of data. To avoid this, the driver should automatically disable DMA for these drives.

- Western Digital WDC AC11000H, AC22100H, AC32500H, AC33100H, AC31600H − all versions
- Western Digital WDC AC32100H revision 24.09P07
- Western Digital WDC AC23200L revision 21.10N21

9.2 Are you overclocking?

If you are, beware! Here is a quote from the old udma−generic documentation:

```
DON'T OVERCLOCK the PCI bus. 37.5MHz is the maximum supported speed for
the PCI bus. Some (supposedly compatible) UDMA drives will not even take
37.5MHz, but should be OK at 33.3MHz.

In any case, NEVER, NEVER set the PCI bus to 41.5MHz.

The RECOMMENDED safe setting is 33MHz.
```

9.3 Is your BIOS current?

Here is another clip from the udma−generic docs:

```
The real work involved in setting up the chips for DMA transfers is done
mostly by the BIOS of each motherboard. Now of course one hopes that the
BIOS has been correctly programmed...

For example, the ASUS SP-97V motherboard with its original BIOS (Rev. 1.03)
would malfunction with the modified Linux driver in both DMA mode 2 and UDMA
modes; it would work well using PIO mode 4, or under Windows 95 in all
modes. I downloaded the latest BIOS image (Rev. 1.06) from the ASUS Web site
and flashed the BIOS EPROM with the latest BIOS revision. It has been
working perfectly ever since (at 66 MHz bus speeds).

What this tells us is that the BIOS sets up the DMA controller with specific
timing parameters (active pulse and recovery clock cycles). My initial BIOS
revision probably had bad timings. Since the Windows 95 driver sets up those
timings by itself (i.e. it does not depend on the BIOS to setup the hard
disk controller timing parameters), I initially had problems only with the
Linux driver, while Windows 95 worked well.

So, let me state this again: this Linux (U)DMA driver depends on the BIOS for
correct (U)DMA controller setup. If you have problems, first check that you
have the latest BIOS revision for your specific motherboard.

...

New BIOS revisions can be downloaded from your motherboard manufacturer's
Web site. Flashing a new BIOS image is a simple operation but one must
strictly follow the steps explained on the motherboard manual.

Late Award BIOS revisions seem stable with respect to UDMA. Anything with a
date of 1998 should be fine.
```

9.4 If you still can't get it to work!

If nothing in this document proved helpful, or at least not helpful enough to get your machine working, your best bet is to write up a message that fully describes your difficulty, what type of UDMA interface you have, whether it is onboard or on a card, if your drive is actually UDMA or plain EIDE, exactly what configuration of drives you have, what version (distribution & kernel versions if possible) of Linux you are using, and anything else that sounds useful, and post it to the newsgroup comp.os.linux.hardware. You will probably get some helpful suggestions soon.

---

10. If you have some information about UDMA stuff that's not in this mini−howto...

Great! If you know something I don't, by all means send it to me (brion@pobox.com) and I will put it in this document and update it fairly soon.
