Linux M68k Documentation

The kernel development community

Jan 15, 2023

CONTENTS

CHAPTER

ONE

COMMAND LINE OPTIONS FOR LINUX/M68K

Last Update: 2 May 1999

Linux/m68k version: 2.2.6

Author: Roman.Hodek@informatik.uni-erlangen.de (Roman Hodek)

Update: jds@kom.auc.dk (Jes Sorensen) and faq@linux-m68k.org (Chris Lawrence)

1.1 0) Introduction

Often I've been asked which command line options the Linux/m68k kernel understands, or how the exact syntax for the ... option is, or ... about the option ... . I hope, this document supplies all the answers...

Note that some options might be outdated, their descriptions being incomplete or missing. Please update the information and send in the patches.

1.2 1) Overview of the Kernel's Option Processing

The kernel knows three kinds of options on its command line:

1) kernel options
2) environment settings
3) arguments for init

To which of these classes an argument belongs is determined as follows: If the option is known to the kernel itself, i.e. if the name (the part before the '=') or, in some cases, the whole argument string is known to the kernel, it belongs to class 1. Otherwise, if the argument contains an '=', it is of class 2, and the definition is put into init's environment. All other arguments are passed to init as command line options.

This document describes the valid kernel options for Linux/m68k in the version mentioned at the start of this file. Later revisions may add new such options, and some may be missing in older versions.

In general, the value (the part after the '=') of an option is a list of values separated by commas. The interpretation of these values is up to the driver that "owns" the option. This association of options with drivers is also the reason that some are further subdivided.

1.3 2) General Kernel Options

1.3.1 2.1) root=

Syntax root=/dev/<device>

or root=<hex_number>

This tells the kernel which device it should mount as the root filesystem. The device must be a block device with a valid filesystem on it.

The first syntax gives the device by name. These names are converted into a major/minor number internally in the kernel in an unusual way. Normally, this "conversion" is done by the device files in /dev, but this isn't possible here, because the root filesystem (with /dev) isn't mounted yet... So the kernel parses the name itself, with some hardcoded name to number mappings. The name must always be a combination of two or three letters, followed by a decimal number. Valid names are:

```
/dev/ram: -> 0x0100 (initial ramdisk)
/dev/hda: -> 0x0300 (first IDE disk)
/dev/hdb: -> 0x0340 (second IDE disk)
/dev/sda: -> 0x0800 (first SCSI disk)
/dev/sdb: -> 0x0810 (second SCSI disk)
/dev/sdc: -> 0x0820 (third SCSI disk)
/dev/sdd: -> 0x0830 (forth SCSI disk)
/dev/sde: -> 0x0840 (fifth SCSI disk)
/dev/fd : -> 0x0200 (floppy disk)
```

The name must be followed by a decimal number, that stands for the partition number. Internally, the value of the number is just added to the device number mentioned in the table above. The exceptions are /dev/ram and /dev/fd, where /dev/ram refers to an initial ramdisk loaded by your bootstrap program (please consult the instructions for your bootstrap program to find out how to load an initial ramdisk). As of kernel version 2.0.18 you must specify /dev/ram as the root device if you want to boot from an initial ramdisk. For the floppy devices, /dev/fd, the number stands for the floppy drive number (there are no partitions on floppy disks). I.e., /dev/fd0 stands for the first drive, /dev/fd1 for the second, and so on. Since the number is just added, you can also force the disk format by adding a number greater than 3. If you look into your /dev directory, use can see the /dev/fd0D720 has major 2 and minor 16. You can specify this device for the root FS by writing "root=/dev/fd16" on the kernel command line.

[Strange and maybe uninteresting stuff ON]

This unusual translation of device names has some strange consequences: If, for example, you have a symbolic link from /dev/fd to /dev/fd0D720 as an abbreviation for floppy driver #0 in DD format, you cannot use this name for specifying the root device, because the kernel cannot see this symlink before mounting the root FS and it isn't in the table above. If you use it, the root device will not be set at all, without an error message. Another example: You cannot use a partition on e.g. the sixth SCSI disk as the root filesystem, if you want to specify it by name. This is, because only the devices up to /dev/sde are in the table above, but not /dev/sdf. Although, you can use the sixth SCSI disk for the root FS, but you have to specify the device by number... (see below). Or, even more strange, you can use the fact that there is no range checking of the partition number, and your knowledge that each disk uses 16 minors, and write "root=/dev/sde17" (for /dev/sdf1).

[Strange and maybe uninteresting stuff OFF]

If the device containing your root partition isn't in the table above, you can also specify it by major and minor numbers. These are written in hex, with no prefix and no separator between. E.g., if you have a CD with contents appropriate as a root filesystem in the first SCSI CD-ROM drive, you boot from it by "root=0b00". Here, hex "0b" = decimal 11 is the major of SCSI CD-ROMs, and the minor 0 stands for the first of these. You can find out all valid major numbers by looking into include/linux/major.h.

In addition to major and minor numbers, if the device containing your root partition uses a partition table format with unique partition identifiers, then you may use them. For instance, "root=PARTUUID=00112233-4455-6677-8899-AABBCCDDEEFF". It is also possible to reference another partition on the same device using a known partition UUID as the starting point. For example, if partition 5 of the device has the UUID of 00112233-4455-6677-8899-AABBCCDDEEFF then partition 3 may be found as follows:

PARTUUID=00112233-4455-6677-8899-AABBCCDDEEFF/PARTNROFF=-2

Authoritative information can be found in "Documentation/admin-guide/kernel-parameters.rst".

1.3.2 2.2) ro, rw

Syntax ro

or rw

These two options tell the kernel whether it should mount the root filesystem read-only or read-write. The default is read-only, except for ramdisks, which default to read-write.

1.3.3 2.3) debug

Syntax debug

This raises the kernel log level to 10 (the default is 7). This is the same level as set by the "dmesg" command, just that the maximum level selectable by dmesg is 8.

1.3.4 2.4) debug=

Syntax debug=<device>

This option causes certain kernel messages be printed to the selected debugging device. This can aid debugging the kernel, since the messages can be captured and analyzed on some other machine. Which devices are possible depends on the machine type. There are no checks for the validity of the device name. If the device isn't implemented, nothing happens.

Messages logged this way are in general stack dumps after kernel memory faults or bad kernel traps, and kernel panics. To be exact: all messages of level 0 (panic messages) and all messages printed while the log level is 8 or more (their level doesn't matter). Before stack dumps, the kernel sets the log level to 10 automatically. A level of at least 8 can also be set by the "debug" command line option (see 2.3) and at run time with "dmesg -n 8".

Devices possible for Amiga:

- **"ser":** built-in serial port; parameters: 9600bps, 8N1

- **“mem”:** Save the messages to a reserved area in chip mem. After rebooting, they can be read under AmigaOS with the tool ‘dmesg’.

Devices possible for Atari:

- **“ser1”:** ST-MFP serial port (“Modem1”); parameters: 9600bps, 8N1
- **“ser2”:** SCC channel B serial port (“Modem2”); parameters: 9600bps, 8N1
- **“ser” :** default serial port This is “ser2” for a Falcon, and “ser1” for any other machine
- **“midi”:** The MIDI port; parameters: 31250bps, 8N1
- **“par” :** parallel port

  The printing routine for this implements a timeout for the case there’s no printer connected (else the kernel would lock up). The timeout is not exact, but usually a few seconds.

1.3.5 2.6) ramdisk_size=

Syntax ramdisk_size=<size>

This option instructs the kernel to set up a ramdisk of the given size in KBytes. Do not use this option if the ramdisk contents are passed by bootstrap! In this case, the size is selected automatically and should not be overwritten.

The only application is for root filesystems on floppy disks, that should be loaded into memory. To do that, select the corresponding size of the disk as ramdisk size, and set the root device to the disk drive (with “root=”).

2.7) swap=

I can’t find any sign of this option in 2.2.6.

1.3.6 2.8) buff=

I can’t find any sign of this option in 2.2.6.

1.4 3) General Device Options (Amiga and Atari)

1.4.1 3.1) ether=

Syntax ether=[<irq>[,<base_addr>[,<mem_start>[,<mem_end>]]]],<dev-name>

<dev-name> is the name of a net driver, as specified in drivers/net/Space.c in the Linux source. Most prominent are eth0, … eth3, sl0, … sl3, ppp0, …, ppp3, dummy, and lo.

The non-ethernet drivers (sl, ppp, dummy, lo) obviously ignore the settings by this options. Also, the existing ethernet drivers for Linux/m68k (ariadne, a2065, hydra) don’t use them because Zorro boards are really Plug-‘n-Play, so the “ether=” option is useless altogether for Linux/m68k.

1.4.2 3.2) hd=

Syntax hd=<cylinders>,<heads>,<sectors>

This option sets the disk geometry of an IDE disk. The first hd= option is for the first IDE disk, the second for the second one. (I.e., you can give this option twice.) In most cases, you won't have to use this option, since the kernel can obtain the geometry data itself. It exists just for the case that this fails for one of your disks.

1.4.3 3.3) max_scsi_luns=

Syntax max_scsi_luns=<n>

Sets the maximum number of LUNs (logical units) of SCSI devices to be scanned. Valid values for <n> are between 1 and 8. Default is 8 if "Probe all LUNs on each SCSI device" was selected during the kernel configuration, else 1.

1.4.4 3.4) st=

Syntax st=<buffer_size>,[<write_thres>,[<max_buffers>]]

Sets several parameters of the SCSI tape driver. <buffer_size> is the number of 512-byte buffers reserved for tape operations for each device. <write_thres> sets the number of blocks which must be filled to start an actual write operation to the tape. Maximum value is the total number of buffers. <max_buffer> limits the total number of buffers allocated for all tape devices.

1.4.5 3.5) dmasound=

Syntax dmasound=[<buffers>,<buffer-size>[,<catch-radius>]]

This option controls some configurations of the Linux/m68k DMA sound driver (Amiga and Atari): <buffers> is the number of buffers you want to use (minimum 4, default 4), <buffer-size> is the size of each buffer in kilobytes (minimum 4, default 32) and <catch-radius> says how much percent of error will be tolerated when setting a frequency (maximum 10, default 0). For example with 3% you can play 8000Hz AU-Files on the Falcon with its hardware frequency of 8195Hz and thus don't need to expand the sound.

1.5 4) Options for Atari Only

1.5.1 4.1) video=

Syntax video=<fbname>:<sub-options...>

The <fbname> parameter specifies the name of the frame buffer, eg. most atari users will want to specify *atafb* here. The <sub-options> is a comma-separated list of the sub-options listed below.

NB: Please notice that this option was renamed from *atavideo* to *video* during the development of the 1.3.x kernels, thus you might need to update your boot-scripts if upgrading to 2.x from an 1.2.x kernel.

NBB: The behavior of video= was changed in 2.1.57 so the recommended option is to specify the name of the frame buffer.

1.5.2 4.1.1) Video Mode

This sub-option may be any of the predefined video modes, as listed in atari/atafb.c in the Linux/m68k source tree. The kernel will activate the given video mode at boot time and make it the default mode, if the hardware allows. Currently defined names are:

- stlow : 320x200x4
- stmid, default5 : 640x200x2
- sthigh, default4: 640x400x1
- ttlow : 320x480x8, TT only
- ttmid, default1 : 640x480x4, TT only
- tthigh, default2: 1280x960x1, TT only
- vga2 : 640x480x1, Falcon only
- vga4 : 640x480x2, Falcon only
- vga16, default3 : 640x480x4, Falcon only
- vga256 : 640x480x8, Falcon only
- falh2 : 896x608x1, Falcon only
- falh16 : 896x608x4, Falcon only

If no video mode is given on the command line, the kernel tries the modes names “default<n>” in turn, until one is possible with the hardware in use.

A video mode setting doesn’t make sense, if the external driver is activated by a “external:” sub-option.

1.5.3 4.1.2) inverse

Invert the display. This affects both, text (consoles) and graphics (X) display. Usually, the background is chosen to be black. With this option, you can make the background white.

1.5.4 4.1.3) font

Syntax font:<fontname>

Specify the font to use in text modes. Currently you can choose only between *VGA8x8*, *VGA8x16* and *PEARL8x8*. *VGA8x8* is default, if the vertical size of the display is less than 400 pixel rows. Otherwise, the *VGA8x16* font is the default.

1.5.5 4.1.4) *hwscroll_*

Syntax *hwscroll_<n>*

The number of additional lines of video memory to reserve for speeding up the scrolling ("hardware scrolling"). Hardware scrolling is possible only if the kernel can set the video base address in steps fine enough. This is true for STE, MegaSTE, TT, and Falcon. It is not possible with plain STs and graphics cards (The former because the base address must be on a 256 byte boundary there, the latter because the kernel doesn't know how to set the base address at all.)

By default, <n> is set to the number of visible text lines on the display. Thus, the amount of video memory is doubled, compared to no hardware scrolling. You can turn off the hardware scrolling altogether by setting <n> to 0.

1.5.6 4.1.5) internal:

Syntax internal:<xres>;<yres>[;<xres_max>;<yres_max>;<offset>]

This option specifies the capabilities of some extended internal video hardware, like e.g. OverScan. <xres> and <yres> give the (extended) dimensions of the screen.

If your OverScan needs a black border, you have to write the last three arguments of the "internal:". <xres_max> is the maximum line length the hardware allows, <yres_max> the maximum number of lines. <offset> is the offset of the visible part of the screen memory to its physical start, in bytes.

Often, extended interval video hardware has to be activated somehow. For this, see the "sw_*" options below.

1.5.7 4.1.6) external:

Syntax external:<xres>;<yres>;<depth>;<org>;<scrmem>[;<scrlen>[;<vgabase>[;<colw>[;<coltype>[;<xres_virtual>]]]]]

This is probably the most complicated parameter... It specifies that you have some external video hardware (a graphics board), and how to use it under Linux/m68k. The kernel cannot know more about the hardware than you tell it here! The kernel also is unable to set or change any video modes, since it doesn't know about any board internal. So, you have to switch to that video mode before you start Linux, and cannot switch to another mode once Linux has started.

The first 3 parameters of this sub-option should be obvious: <xres>, <yres> and <depth> give the dimensions of the screen and the number of planes (depth). The depth is the logarithm to base 2 of the number of colors possible. (Or, the other way round: The number of colors is 2^depth).

You have to tell the kernel furthermore how the video memory is organized. This is done by a letter as <org> parameter:

'n': "normal planes", i.e. one whole plane after another

'i': "interleaved planes", i.e. 16 bit of the first plane, than 16 bit of the next, and so on... This mode is used only with the built-in Atari video modes, I think there is no card that supports this mode.

'p': "packed pixels", i.e. <depth> consecutive bits stand for all planes of one pixel; this is the most common mode for 8 planes (256 colors) on graphic cards

't': "true color" (more or less packed pixels, but without a color lookup table); usually depth is 24

For monochrome modes (i.e., <depth> is 1), the <org> letter has a different meaning:

'n': normal colors, i.e. 0=white, 1=black

'i': inverted colors, i.e. 0=black, 1=white

The next important information about the video hardware is the base address of the video memory. That is given in the <scrmem> parameter, as a hexadecimal number with a "0x" prefix. You have to find out this address in the documentation of your hardware.

The next parameter, <scrlen>, tells the kernel about the size of the video memory. If it's missing, the size is calculated from <xres>, <yres>, and <depth>. For now, it is not useful to write a value here. It would be used only for hardware scrolling (which isn't possible with the external driver, because the kernel cannot set the video base address), or for virtual resolutions under X (which the X server doesn't support yet). So, it's currently best to leave this field empty, either by ending the "external:" after the video address or by writing two consecutive semicolons, if you want to give a <vgabase> (it is allowed to leave this parameter empty).

The <vgabase> parameter is optional. If it is not given, the kernel cannot read or write any color registers of the video hardware, and thus you have to set appropriate colors before you start Linux. But if your card is somehow VGA compatible, you can tell the kernel the base address of the VGA register set, so it can change the color lookup table. You have to look up this address in your board's documentation. To avoid misunderstandings: <vgabase> is the _base_ address, i.e. a 4k aligned address. For read/writing the color registers, the kernel uses the addresses vgabase+0x3c7...vgabase+0x3c9. The <vgabase> parameter is written in hexadecimal with a "0x" prefix, just as <scrmem>.

<colw> is meaningful only if <vgabase> is specified. It tells the kernel how wide each of the color register is, i.e. the number of bits per single color (red/green/blue). Default is 6, another quite usual value is 8.

Also <coltype> is used together with <vgabase>. It tells the kernel about the color register model of your gfx board. Currently, the types "vga" (which is also the default) and "mv300" (SANG MV300) are implemented.

Parameter <xres_virtual> is required for ProMST or ET4000 cards where the physical line-length differs from the visible length. With ProMST, xres_virtual must be set to 2048. For ET4000, xres_virtual depends on the initialisation of the video-card. If you're missing a corresponding yres_virtual: the external part is legacy, therefore we don't support hardware-dependent functions like hardware-scroll, panning or blanking.

1.5.8 4.1.7) eclock:

The external pixel clock attached to the Falcon VIDEL shifter. This currently works only with the ScreenWonder!

1.5.9 4.1.8) monitorcap:

Syntax monitorcap:<vmin>;<vmax>;<hmin>;<hmax>

This describes the capabilities of a multisync monitor. Don't use it with a fixed-frequency monitor! For now, only the Falcon frame buffer uses the settings of "monitorcap:".

<vmin> and <vmax> are the minimum and maximum, resp., vertical frequencies your monitor can work with, in Hz. <hmin> and <hmax> are the same for the horizontal frequency, in kHz.

The defaults are 58;62;31;32 (VGA compatible).

The defaults for TV/SC1224/SC1435 cover both PAL and NTSC standards.

1.5.10 4.1.9) keep

If this option is given, the framebuffer device doesn't do any video mode calculations and settings on its own. The only Atari fb device that does this currently is the Falcon.

What you reach with this: Settings for unknown video extensions aren't overridden by the driver, so you can still use the mode found when booting, when the driver doesn't know to set this mode itself. But this also means, that you can't switch video modes anymore...

An example where you may want to use "keep" is the ScreenBlaster for the Falcon.

1.5.11 4.2) atamouse=

Syntax atamouse=<x-threshold>,[<y-threshold>]

With this option, you can set the mouse movement reporting threshold. This is the number of pixels of mouse movement that have to accumulate before the IKBD sends a new mouse packet to the kernel. Higher values reduce the mouse interrupt load and thus reduce the chance of keyboard overruns. Lower values give a slightly faster mouse responses and slightly better mouse tracking.

You can set the threshold in x and y separately, but usually this is of little practical use. If there's just one number in the option, it is used for both dimensions. The default value is 2 for both thresholds.

1.5.12 4.3) ataflop=

Syntax ataflop=<drive type>[,<trackbuffering>[,<steprateA>[,<steprateB>]]]

The drive type may be 0, 1, or 2, for DD, HD, and ED, resp. This setting affects how many buffers are reserved and which formats are probed (see also below). The default is 1 (HD). Only one drive type can be selected. If you have two disk drives, select the "better" type.

The second parameter <trackbuffer> tells the kernel whether to use track buffering (1) or not (0). The default is machine-dependent: no for the Medusa and yes for all others.

With the two following parameters, you can change the default steprate used for drive A and B, resp.

1.5.13 4.4) atascsi=

Syntax atascsi=<can_queue>[,<cmd_per_lun>[,<scat-gat>[,<host-id>[,<tagged>]]]]

This option sets some parameters for the Atari native SCSI driver. Generally, any number of arguments can be omitted from the end. And for each of the numbers, a negative value means "use default". The defaults depend on whether TT-style or Falcon-style SCSI is used. Below, defaults are noted as n/m, where the first value refers to TT-SCSI and the latter to Falcon-SCSI. If an illegal value is given for one parameter, an error message is printed and that one setting is ignored (others aren't affected).

<can_queue>: This is the maximum number of SCSI commands queued internally to the Atari SCSI driver. A value of 1 effectively turns off the driver internal multitasking (if it causes problems). Legal values are >= 1. <can_queue> can be as high as you like, but values greater than <cmd_per_lun> times the number of SCSI targets (LUNs) you have don't make sense. Default: 16/8.

<cmd_per_lun>: Maximum number of SCSI commands issued to the driver for one logical unit (LUN, usually one SCSI target). Legal values start from 1. If tagged queuing (see below) is not used, values greater than 2 don't make sense, but waste memory. Otherwise, the maximum is the number of command tags available to the driver (currently 32). Default: 8/1. (Note: Values > 1 seem to cause problems on a Falcon, cause not yet known.)

The <cmd_per_lun> value at a great part determines the amount of memory SCSI reserves for itself. The formula is rather complicated, but I can give you some hints:

no scatter-gather: cmd_per_lun * 232 bytes

full scatter-gather: cmd_per_lun * approx. 17 Kbytes

<scat-gat>: Size of the scatter-gather table, i.e. the number of requests consecutive on the disk that can be merged into one SCSI command. Legal values are between 0 and 255. Default: 255/0. Note: This value is forced to 0 on a Falcon, since scatter-gather isn't possible with the ST-DMA. Not using scatter-gather hurts performance significantly.

<host-id>: The SCSI ID to be used by the initiator (your Atari). This is usually 7, the highest possible ID. Every ID on the SCSI bus must be unique. Default: determined at run time: If the NV-RAM checksum is valid, and bit 7 in byte 30 of the NV-RAM is set, the lower 3 bits of this byte are used as the host ID. (This method is defined by Atari and also used by some TOS HD drivers.) If the above isn't given, the default ID is 7. (both, TT and Falcon).

<tagged>: 0 means turn off tagged queuing support, all other values > 0 mean use tagged queuing for targets that support it. Default: currently off, but this may change when tagged queuing handling has been proved to be reliable.

Tagged queuing means that more than one command can be issued to one LUN, and the SCSI device itself orders the requests so they can be performed in optimal order. Not all SCSI devices support tagged queuing (:-().

1.5.14 4.5 switches=

Syntax switches=<list of switches>

With this option you can switch some hardware lines that are often used to enable/disable certain hardware extensions. Examples are OverScan, overclocking, ...

The <list of switches> is a comma-separated list of the following items:

ikbd: set RTS of the keyboard ACIA high

midi: set RTS of the MIDI ACIA high

snd6: set bit 6 of the PSG port A

snd7: set bit 6 of the PSG port A

It doesn't make sense to mention a switch more than once (no difference to only once), but you can give as many switches as you want to enable different features. The switch lines are set as early as possible during kernel initialization (even before determining the present hardware.)

All of the items can also be prefixed with *ov_*, i.e. *ov_ikbd*, *ov_midi*, ... These options are meant for switching on an OverScan video extension. The difference to the bare option is that the switch-on is done after video initialization, and somehow synchronized to the HBLANK. A speciality is that ov_ikbd and ov_midi are switched off before rebooting, so that OverScan is disabled and TOS boots correctly.

If you give an option both, with and without the *ov_* prefix, the earlier initialization (*ov_*-less) takes precedence. But the switching-off on reset still happens in this case.

1.6 5) Options for Amiga Only:

1.6.1 5.1) video=

Syntax video=<fbname>:<sub-options...>

The <fbname> parameter specifies the name of the frame buffer, valid options are *amifb*, *cyber*, 'virge', *retz3* and *clgen*, provided that the respective frame buffer devices have been compiled into the kernel (or compiled as loadable modules). The behavior of the <fbname> option was changed in 2.1.57 so it is now recommended to specify this option.

The <sub-options> is a comma-separated list of the sub-options listed below. This option is organized similar to the Atari version of the “video”-option (4.1), but knows fewer sub-options.

1.6.2 5.1.1) video mode

Again, similar to the video mode for the Atari (see 4.1.1). Predefined modes depend on the used frame buffer device.

OCS, ECS and AGA machines all use the color frame buffer. The following predefined video modes are available:

NTSC modes:

- ntsc : 640x200, 15 kHz, 60 Hz
- ntsc-lace : 640x400, 15 kHz, 60 Hz interlaced

PAL modes:

- pal : 640x256, 15 kHz, 50 Hz
- pal-lace : 640x512, 15 kHz, 50 Hz interlaced

ECS modes:

- multiscan : 640x480, 29 kHz, 57 Hz
- multiscan-lace : 640x960, 29 kHz, 57 Hz interlaced
- euro36 : 640x200, 15 kHz, 72 Hz
- euro36-lace : 640x400, 15 kHz, 72 Hz interlaced
- euro72 : 640x400, 29 kHz, 68 Hz
- euro72-lace : 640x800, 29 kHz, 68 Hz interlaced
- super72 : 800x300, 23 kHz, 70 Hz
- super72-lace : 800x600, 23 kHz, 70 Hz interlaced
- dblntsc-ff : 640x400, 27 kHz, 57 Hz
- dblntsc-lace : 640x800, 27 kHz, 57 Hz interlaced
- dblpal-ff : 640x512, 27 kHz, 47 Hz
- dblpal-lace : 640x1024, 27 kHz, 47 Hz interlaced
- dblntsc : 640x200, 27 kHz, 57 Hz doublescan
- dblpal : 640x256, 27 kHz, 47 Hz doublescan

VGA modes:

- vga : 640x480, 31 kHz, 60 Hz
- vga70 : 640x400, 31 kHz, 70 Hz

Please notice that the ECS and VGA modes require either an ECS or AGA chipset, and that these modes are limited to 2-bit color for the ECS chipset and 8-bit color for the AGA chipset.

1.6.3 5.1.2) depth

Syntax depth:<nr. of bit-planes>

Specify the number of bit-planes for the selected video-mode.

1.6.4 5.1.3) inverse

Use inverted display (black on white). Functionally the same as the “inverse” sub-option for the Atari.

1.6.5 5.1.4) font

Syntax font:<fontname>

Specify the font to use in text modes. Functionally the same as the “font” sub-option for the Atari, except that *PEARL8x8* is used instead of *VGA8x8* if the vertical size of the display is less than 400 pixel rows.

1.6.6 5.1.5) monitorcap:

Syntax monitorcap:<vmin>;<vmax>;<hmin>;<hmax>

This describes the capabilities of a multisync monitor. For now, only the color frame buffer uses the settings of “monitorcap:”.

<vmin> and <vmax> are the minimum and maximum, resp., vertical frequencies your monitor can work with, in Hz. <hmin> and <hmax> are the same for the horizontal frequency, in kHz.

The defaults are 50;90;15;38 (Generic Amiga multisync monitor).

1.6.7 5.2) fd_def_df0=

Syntax fd_def_df0=<value>

Sets the df0 value for “silent” floppy drives. The value should be in hexadecimal with “0x” prefix.

1.6.8 5.3) wd33c93=

Syntax wd33c93=<sub-options...>

These options affect the A590/A2091, A3000 and GVP Series II SCSI controllers.

The <sub-options> is a comma-separated list of the sub-options listed below.

1.6.9 5.3.1) nosync

Syntax nosync:bitmask

bitmask is a byte where the 1st 7 bits correspond with the 7 possible SCSI devices. Set a bit to prevent sync negotiation on that device. To maintain backwards compatibility, a command-line such as "wd33c93=255" will be automatically translated to "wd33c93=nosync:0xff". The default is to disable sync negotiation for all devices, eg. nosync:0xff.

1.6.10 5.3.2) period

Syntax period:ns

ns is the minimum # of nanoseconds in a SCSI data transfer period. Default is 500; acceptable values are 250 - 1000.

1.6.11 5.3.3) disconnect

Syntax disconnect:x

Specify x = 0 to never allow disconnects, 2 to always allow them. x = 1 does 'adaptive' disconnects, which is the default and generally the best choice.

1.6.12 5.3.4) debug

Syntax debug:x

If *DEBUGGING_ON* is defined, x is a bit mask that causes various types of debug output to printed - see the DB_xxx defines in wd33c93.h.

1.6.13 5.3.5) clock

Syntax clock:x

x = clock input in MHz for WD33c93 chip. Normal values would be from 8 through 20. The default value depends on your hostadapter(s), default for the A3000 internal controller is 14, for the A2091 it's 8 and for the GVP hostadapters it's either 8 or 14, depending on the hostadapter and the SCSI-clock jumper present on some GVP hostadapters.

1.6.14 5.3.6) next

No argument. Used to separate blocks of keywords when there's more than one wd33c93-based host adapter in the system.

1.6.15 5.3.7) nodma

Syntax nodma:x

If x is 1 (or if the option is just written as “nodma”), the WD33c93 controller will not use DMA (= direct memory access) to access the Amiga’s memory. This is useful for some systems (like A3000’s and A4000’s with the A3640 accelerator, revision 3.0) that have problems using DMA to chip memory. The default is 0, i.e. to use DMA if possible.

1.6.16 5.4) gvp11=

Syntax gvp11=<addr-mask>

The earlier versions of the GVP driver did not handle DMA address-mask settings correctly which made it necessary for some people to use this option, in order to get their GVP controller running under Linux. These problems have hopefully been solved and the use of this option is now highly unrecommended!

Incorrect use can lead to unpredictable behavior, so please only use this option if you *know* what you are doing and have a reason to do so. In any case if you experience problems and need to use this option, please inform us about it by mailing to the Linux/68k kernel mailing list.

The address mask set by this option specifies which addresses are valid for DMA with the GVP Series II SCSI controller. An address is valid, if no bits are set except the bits that are set in the mask, too.

Some versions of the GVP can only DMA into a 24 bit address range, some can address a 25 bit address range while others can use the whole 32 bit address range for DMA. The correct setting depends on your controller and should be autodetected by the driver. An example is the 24 bit region which is specified by a mask of 0x00fffffe.

CHAPTER

TWO

AMIGA BUDDHA AND CATWEASEL IDE DRIVER

The Amiga Buddha and Catweasel IDE Driver (part of ide.c) was written by Geert Uytterhoeven based on the following specifications:

Register map of the Buddha IDE controller and the Buddha-part of the Catweasel Zorro-II version

The Autoconfiguration has been implemented just as Commodore described in their manuals, no tricks have been used (for example leaving some address lines out of the equations...). If you want to configure the board yourself (for example let a Linux kernel configure the card), look at the Commodore Docs. Reading the nibbles should give this information:

```
Vendor number: 4626 ($1212)
product number: 0 (42 for Catweasel Z-II)
Serial number: 0
Rom-vector: $1000
```

The card should be a Z-II board, size 64K, not for freemem list, Rom-Vektor is valid, no second Autoconfig-board on the same card, no space preference, supports “Shutup_forever”.

Setting the base address should be done in two steps, just as the Amiga Kickstart does: The lower nibble of the 8-Bit address is written to $4a, then the whole Byte is written to $48, while it doesn’t matter how often you’re writing to $4a as long as $48 is not touched. After $48 has been written, the whole card disappears from $e8 and is mapped to the new address just written. Make sure $4a is written before $48, otherwise your chance is only 1:16 to find the board :-).

The local memory-map is even active when mapped to $e8:

$0-$7e	Autokonfig-space, see Z-II docs.
$80-$7fd	reserved
$7fe	Speed-select Register: Read & Write (description see further down)
$800-$8ff	IDE-Select 0 (Port 0, Register set 0)
$900-$9ff	IDE-Select 1 (Port 0, Register set 1)
$a00-$aff	IDE-Select 2 (Port 1, Register set 0)
$b00-$bff	IDE-Select 3 (Port 1, Register set 1)
$c00-$cff	IDE-Select 4 (Port 2, Register set 0, Catweasel only!)
$d00-$dff	IDE-Select 5 (Port 3, Register set 1, Catweasel only!)
$e00-$eff	local expansion port, on Catweasel Z-II the Catweasel registers are also mapped here. Never touch, use multidisk.device!
$f00	read only, Byte-access: Bit 7 shows the level of the IRQ-line of IDE port 0.
$f01-$f3f	mirror of $f00
$f40	read only, Byte-access: Bit 7 shows the level of the IRQ-line of IDE port 1.
$f41-$f7f	mirror of $f40
$f80	read only, Byte-access: Bit 7 shows the level of the IRQ-line of IDE port 2. (Catweasel only!)
$f81-$fbf	mirror of $f80
$fc0	write-only: Writing any value to this register enables IRQs to be passed from the IDE ports to the Zorro bus. This mechanism has been implemented to be compatible with harddisks that are either defective or have a buggy firmware and pull the IRQ line up while starting up. If interrupts would always be passed to the bus, the computer might not start up. Once enabled, this flag can not be disabled again. The level of the flag can not be determined by software (what for? Write to me if it's necessary!).
$fc1-$fff	mirror of $fc0
$1000-$ffff	Buddha-Rom with offset $1000 in the rom chip. The addresses $0 to $fff of the rom chip cannot be read. Rom is Byte-wide and mapped to even addresses.

The IDE ports issue an INT2. You can read the level of the IRQ-lines of the IDE-ports by reading from the three (two for Buddha-only) registers $f00, $f40 and $f80. This way more than one I/O request can be handled and you can easily determine what driver has to serve the INT2. Buddha and Catweasel expansion boards can issue an INT6. A separate memory map is available for the I/O module and the sysop's I/O module.

The IDE ports are fed by the address lines A2 to A4, just as the Amiga 1200 and Amiga 4000 IDE ports are. This way existing drivers can be easily ported to Buddha. A move.l polls two words out of the same address of IDE port since every word is mirrored once. movem is not possible, but it's not necessary either, because you can only speedup 68000 systems with this technique. A 68020 system with fastmem is faster with move.l.

If you're using the mirrored registers of the IDE-ports with A6=1, the Buddha doesn't care about the speed that you have selected in the speed register (see further down). With A6=1 (for example $840 for port 0, register set 0), a 780ns access is being made. These registers should be used for a command access to the harddisk/CD-Rom, since command accesses are Byte-wide and have to be made slower according to the ATA-X3T9 manual.

Now for the speed-register: The register is byte-wide, and only the upper three bits are used (Bits 7 to 5). Bit 4 must always be set to 1 to be compatible with later Buddha versions (if I'll ever update this one). I presume that I'll never use the lower four bits, but they have to be set to 1 by definition.

The values in this table have to be shifted 5 bits to the left and or'd with $1f (this sets the lower 5 bits).

All the timings have in common: Select and IOR/IOW rise at the same time. IOR and IOW have a propagation delay of about 30ns to the clocks on the Zorro bus, that's why the values are no multiple of 71. One clock-cycle is 71ns long (exactly 70,5 at 14,18 Mhz on PAL systems).

value 0 (Default after reset) 497ns Select (7 clock cycles) , IOR/IOW after 172ns (2 clock cycles) (same timing as the Amiga 1200 does on it's IDE port without accelerator card)

value 1 639ns Select (9 clock cycles), IOR/IOW after 243ns (3 clock cycles)

value 2 781ns Select (11 clock cycles), IOR/IOW after 314ns (4 clock cycles)

value 3 355ns Select (5 clock cycles), IOR/IOW after 101ns (1 clock cycle)

value 4 355ns Select (5 clock cycles), IOR/IOW after 172ns (2 clock cycles)

value 5 355ns Select (5 clock cycles), IOR/IOW after 243ns (3 clock cycles)

value 6 1065ns Select (15 clock cycles), IOR/IOW after 314ns (4 clock cycles)

value 7 355ns Select, (5 clock cycles), IOR/IOW after 101ns (1 clock cycle)

When accessing IDE registers with A6=1 (for example $84x), the timing will always be mode 0 8-bit compatible, no matter what you have selected in the speed register:

781ns select, IOR/IOW after 4 clock cycles (=314ns) aktive.

All the timings with a very short select-signal (the 355ns fast accesses) depend on the accelerator card used in the system: Sometimes two more clock cycles are inserted by the bus interface, making the whole access 497ns long. This doesn't affect the reliability of the controller nor the performance of the card, since this doesn't happen very often.

All the timings are calculated and only confirmed by measurements that allowed me to count the clock cycles. If the system is clocked by an oscillator other than 28,37516 Mhz (for example the NTSC-frequency 28,63636 Mhz), each clock cycle is shortened to a bit less than 70ns (not worth mentioning). You could think of a small performance boost by overclocking the system, but you would either need a multisync monitor, or a graphics card, and your internal diskdrive would go crazy, that's why you shouldn't tune your Amiga this way.

Giving you the possibility to write software that is compatible with both the Buddha and the Catweasel Z-II, The Buddha acts just like a Catweasel Z-II with no device connected to the third IDE-port. The IRQ-register $f80 always shows a "no IRQ here" on the Buddha, and accesses to the third IDE port are going into data's Nirwana on the Buddha.

Jens Schönfeld february 19th, 1997

updated may 27th, 1997

eMail: sysop@nostlgic.tng.oche.de

CHAPTER

THREE

FEATURE STATUS ON M68K ARCHITECTURE

Subsystem	Feature	Kconfig	Status	Desc
core	cBPF-JIT	HAVE_CBPF_JIT	TODO	arch
core	eBPF-JIT	HAVE_EBPF_JIT	TODO	arch
core	generic-idle-thread	GENERIC_SMP_IDLE_THREAD	TODO	arch
core	jump-labels	HAVE_ARCH_JUMP_LABEL	TODO	arch
core	thread-info-in-task	THREAD_INFO_IN_TASK	TODO	arch
core	tracehook	HAVE_ARCH_TRACEHOOK	TODO	arch
debug	debug-vm-pgtable	ARCH_HAS_DEBUG_VM_PGTABLE	TODO	arch
debug	gcov-profile-all	ARCH_HAS_GCOV_PROFILE_ALL	TODO	arch
debug	KASAN	HAVE_ARCH_KASAN	TODO	arch
debug	kcov	ARCH_HAS_KCOV	TODO	arch
debug	kgdb	HAVE_ARCH_KGDB	TODO	arch
debug	kmemleak	HAVE_DEBUG_KMEMLEAK	TODO	arch
debug	kprobes	HAVE_KPROBES	TODO	arch
debug	kprobes-on-ftrace	HAVE_KPROBES_ON_FTRACE	TODO	arch
debug	kretprobes	HAVE_KRETPROBES	TODO	arch
debug	optprobes	HAVE_OPTPROBES	TODO	arch
debug	stackprotector	HAVE_STACKPROTECTOR	TODO	arch
debug	uprobes	ARCH_SUPPORTS_UPROBES	TODO	arch
debug	user-ret-profiler	HAVE_USER_RETURN_NOTIFIER	TODO	arch
io	dma-contiguous	HAVE_DMA_CONTIGUOUS	TODO	arch
locking	cmpxchg-local	HAVE_CMPXCHG_LOCAL	TODO	arch
locking	lockdep	LOCKDEP_SUPPORT	TODO	arch
locking	queued-rwlocks	ARCH_USE_QUEUED_RWLOCKS	TODO	arch
locking	queued-spinlocks	ARCH_USE_QUEUED_SPINLOCKS	TODO	arch
perf	kprobes-event	HAVE_REGS_AND_STACK_ACCESS_API	TODO	arch
perf	perf-regs	HAVE_PERF_REGS	TODO	arch
perf	perf-stackdump	HAVE_PERF_USER_STACK_DUMP	TODO	arch
sched	membarrier-sync-core	ARCH_HAS_MEMBARRIER_SYNC_CORE	TODO	arch
sched	numa-balancing	ARCH_SUPPORTS_NUMA_BALANCING	—	arch
seccomp	seccomp-filter	HAVE_ARCH_SECCOMP_FILTER	TODO	arch
time	arch-tick-broadcast	ARCH_HAS_TICK_BROADCAST	TODO	arch
time	clockevents	!LEGACY_TIMER_TICK	TODO	arch
time	context-tracking	HAVE_CONTEXT_TRACKING	TODO	arch
time	irq-time-acct	HAVE_IRQ_TIME_ACCOUNTING	TODO	arch
time	virt-cpuacct	HAVE_VIRT_CPU_ACCOUNTING	TODO	arch

Table 1 – continued from p

Subsystem	Feature	Kconfig	Status	Desc
vm	batch-unmap-tlb-flush	ARCH_WANT_BATCHED_UNMAP_TLB_FLUSH	—	arch
vm	ELF-ASLR	ARCH_HAS_ELF_RANDOMIZE	TODO	arch
vm	huge-vmap	HAVE_ARCH_HUGE_VMAP	TODO	arch
vm	ioremap_prot	HAVE_IOREMAP_PROT	TODO	arch
vm	PG_uncached	ARCH_USES_PG_UNCACHED	TODO	arch
vm	pte_special	ARCH_HAS_PTE_SPECIAL	TODO	arch
vm	THP	HAVE_ARCH_TRANSPARENT_HUGEPAGE	—	arch