﻿# You may now use double quotes around pathnames, in case

# your pathname includes spaces.

#=======================================================================

# PLUGIN\_CTRL:

# Controls the presence of optional device plugins. These plugins are loaded

# directly with this option and some of them install a config option that is

# only available when the plugin device is loaded. The value "1" means to load

# the plugin and "0" will unload it (if loaded before).

#

# These plugins will be loaded by default (if present): 'biosdev', 'extfpuirq',

# 'gameport', 'iodebug','parallel', 'serial', 'speaker' and 'unmapped'.

#

# These plugins are also supported, but they are usually loaded directly with

# their bochsrc option: 'e1000', 'es1370', 'ne2k', 'pcidev', 'pcipnic', 'sb16',

# 'usb\_ohci', 'usb\_uhci', 'usb\_xhci' and 'voodoo'.

#=======================================================================

#plugin\_ctrl: unmapped=0, e1000=1 # unload 'unmapped' and load 'e1000'

#=======================================================================

# CONFIG\_INTERFACE

#

# The configuration interface is a series of menus or dialog boxes that

# allows you to change all the settings that control Bochs's behavior.

# Depending on the platform there are up to 3 choices of configuration

# interface: a text mode version called "textconfig" and two graphical versions

# called "win32config" and "wx". The text mode version uses stdin/stdout and

# is always compiled in, unless Bochs is compiled for wx only. The choice

# "win32config" is only available on win32 and it is the default there.

# The choice "wx" is only available when you use "--with-wx" on the configure

# command. If you do not write a config\_interface line, Bochs will

# choose a default for you.

#

# NOTE: if you use the "wx" configuration interface, you must also use

# the "wx" display library.

#=======================================================================

#config\_interface: textconfig

#config\_interface: win32config

#config\_interface: wx

#=======================================================================

# DISPLAY\_LIBRARY

#

# The display library is the code that displays the Bochs VGA screen. Bochs

# has a selection of about 10 different display library implementations for

# different platforms. If you run configure with multiple --with-\* options,

# the display\_library command lets you choose which one you want to run with.

# If you do not write a display\_library line, Bochs will choose a default for

# you.

#

# The choices are:

# x use X windows interface, cross platform

# win32 use native win32 libraries

# carbon use Carbon library (for MacOS X)

# macintosh use MacOS pre-10

# amigaos use native AmigaOS libraries

# sdl use SDL 1.2.x library, cross platform

# sdl2 use SDL 2.x library, cross platform

# svga use SVGALIB library for Linux, allows graphics without X11

# term text only, uses curses/ncurses library, cross platform

# rfb provides an interface to AT&T's VNC viewer, cross platform

# vncsrv use LibVNCServer for extended RFB(VNC) support

# wx use wxWidgets library, cross platform

# nogui no display at all

#

# NOTE: if you use the "wx" configuration interface, you must also use

# the "wx" display library.

#

# Specific options:

# Some display libraries now support specific options to control their

# behaviour. These options are supported by more than one display library:

#

# "gui\_debug" - use GTK debugger gui (sdl, sdl2, x) / Win32 debugger gui (sdl,

# sdl2, win32)

# "hideIPS" - disable IPS output in status bar (rfb, sdl, sdl2, vncsrv,

# win32, wx, x)

# "nokeyrepeat" - turn off host keyboard repeat (sdl, sdl2, win32, x)

# "timeout" - time (in seconds) to wait for client (rfb, vncsrv)

#

# See the examples below for other currently supported options.

#=======================================================================

#display\_library: amigaos

#display\_library: carbon

#display\_library: macintosh

#display\_library: nogui

#display\_library: rfb

#display\_library: sdl, options="fullscreen" # startup in fullscreen mode

#display\_library: sdl2, options="fullscreen" # startup in fullscreen mode

#display\_library: term

#display\_library: vncsrv

#display\_library: win32

#display\_library: wx

#display\_library: x

#=======================================================================

# CPU:

# This defines cpu-related parameters inside Bochs:

#

# MODEL:

# Selects CPU configuration to emulate from pre-defined list of all

# supported configurations. When this option is used and the value

# is different from 'bx\_generic', the parameters of the CPUID option

# have no effect anymore.

#

# CPU configurations that can be selected:

# -----------------------------------------------------------------

# pentium Intel Pentium (P54C)

# pentium\_mmx Intel Pentium MMX

# amd\_k6\_2\_chomper AMD-K6(tm) 3D processor (Chomper)

# p2\_klamath Intel Pentium II (Klamath)

# p3\_katmai Intel Pentium III (Katmai)

# p4\_willamette Intel(R) Pentium(R) 4 (Willamette)

# core\_duo\_t2400\_yonah Intel(R) Core(TM) Duo CPU T2400 (Yonah)

# atom\_n270 Intel(R) Atom(TM) CPU N270

# p4\_prescott\_celeron\_336 Intel(R) Celeron(R) 336 (Prescott)

# athlon64\_clawhammer AMD Athlon(tm) 64 Processor 2800+ (Clawhammer)

# athlon64\_venice AMD Athlon(tm) 64 Processor 3000+ (Venice)

# turion64\_tyler AMD Turion(tm) 64 X2 Mobile TL-60 (Tyler)

# phenom\_8650\_toliman AMD Phenom X3 8650 (Toliman)

# core2\_penryn\_t9600 Intel Mobile Core 2 Duo T9600 (Penryn)

# corei5\_lynnfield\_750 Intel(R) Core(TM) i5 750 (Lynnfield)

# corei5\_arrandale\_m520 Intel(R) Core(TM) i5 M 520 (Arrandale)

# zambezi AMD FX(tm)-4100 Quad-Core Processor (Zambezi)

# trinity\_apu AMD A8-5600K APU (Trinity)

# corei7\_sandy\_bridge\_2600k Intel(R) Core(TM) i7-2600K (Sandy Bridge)

# corei7\_ivy\_bridge\_3770k Intel(R) Core(TM) i7-3770K CPU (Ivy Bridge)

# corei7\_haswell\_4770 Intel(R) Core(TM) i7-4770 CPU (Haswell)

# broadwell\_ult Intel(R) Processor 5Y70 CPU (Broadwell)

#

# COUNT:

# Set the number of processors:cores per processor:threads per core when

# Bochs is compiled for SMP emulation. Bochs currently supports up to

# 14 threads (legacy APIC) or 254 threads (xAPIC or higher) running

# simultaniosly. If Bochs is compiled without SMP support, it won't accept

# values different from 1.

#

# QUANTUM:

# Maximum amount of instructions allowed to execute by processor before

# returning control to another cpu. This option exists only in Bochs

# binary compiled with SMP support.

#

# RESET\_ON\_TRIPLE\_FAULT:

# Reset the CPU when triple fault occur (highly recommended) rather than

# PANIC. Remember that if you trying to continue after triple fault the

# simulation will be completely bogus !

#

# CPUID\_LIMIT\_WINNT:

# Determine whether to limit maximum CPUID function to 2. This mode is

# required to workaround WinNT installation and boot issues.

#

# MSRS:

# Define path to user CPU Model Specific Registers (MSRs) specification.

# See example in msrs.def.

#

# IGNORE\_BAD\_MSRS:

# Ignore MSR references that Bochs does not understand; print a warning

# message instead of generating #GP exception. This option is enabled

# by default but will not be avaiable if configurable MSRs are enabled.

#

# MWAIT\_IS\_NOP:

# When this option is enabled MWAIT will not put the CPU into a sleep state.

# This option exists only if Bochs compiled with --enable-monitor-mwait.

#

# IPS:

# Emulated Instructions Per Second. This is the number of IPS that bochs

# is capable of running on your machine. You can recompile Bochs with

# --enable-show-ips option enabled, to find your host's capability.

# Measured IPS value will then be logged into your log file or shown

# in the status bar (if supported by the gui).

#

# IPS is used to calibrate many time-dependent events within the bochs

# simulation. For example, changing IPS affects the frequency of VGA

# updates, the duration of time before a key starts to autorepeat, and

# the measurement of BogoMips and other benchmarks.

#

# Examples:

#

# Bochs Machine/Compiler Mips

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 2.4.6 3.4Ghz Intel Core i7 2600 with Win7x64/g++ 4.5.2 85 to 95 Mips

# 2.3.7 3.2Ghz Intel Core 2 Q9770 with WinXP/g++ 3.4 50 to 55 Mips

# 2.3.7 2.6Ghz Intel Core 2 Duo with WinXP/g++ 3.4 38 to 43 Mips

# 2.2.6 2.6Ghz Intel Core 2 Duo with WinXP/g++ 3.4 21 to 25 Mips

# 2.2.6 2.1Ghz Athlon XP with Linux 2.6/g++ 3.4 12 to 15 Mips

#=======================================================================

cpu: model=core2\_penryn\_t9600, count=1, ips=50000000, reset\_on\_triple\_fault=1, ignore\_bad\_msrs=1, msrs="msrs.def"

cpu: cpuid\_limit\_winnt=0

#=======================================================================

# CPUID:

#

# This defines features and functionality supported by Bochs emulated CPU.

# The option has no offect if CPU model was selected in CPU option.

#

# MMX:

# Select MMX instruction set support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 5.

#

# APIC:

# Select APIC configuration (LEGACY/XAPIC/XAPIC\_EXT/X2APIC).

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 5.

#

# SEP:

# Select SYSENTER/SYSEXIT instruction set support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# SIMD:

# Select SIMD instructions support.

# Any of NONE/SSE/SSE2/SSE3/SSSE3/SSE4\_1/SSE4\_2/AVX/AVX2/AVX512

# could be selected.

#

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

# The AVX choises exists only if Bochs compiled with --enable-avx option.

#

# SSE4A:

# Select AMD SSE4A instructions support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# MISALIGNED\_SSE:

# Select AMD Misaligned SSE mode support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# AES:

# Select AES instruction set support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# SHA:

# Select SHA instruction set support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# MOVBE:

# Select MOVBE Intel(R) Atom instruction support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# ADX:

# Select ADCX/ADOX instructions support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# XSAVE:

# Select XSAVE extensions support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# XSAVEOPT:

# Select XSAVEOPT instruction support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# AVX\_F16C:

# Select AVX float16 convert instructions support.

# This option exists only if Bochs compiled with --enable-avx option.

#

# AVX\_FMA:

# Select AVX fused multiply add (FMA) instructions support.

# This option exists only if Bochs compiled with --enable-avx option.

#

# BMI:

# Select BMI1/BMI2 instructions support.

# This option exists only if Bochs compiled with --enable-avx option.

#

# XOP:

# Select AMD XOP instructions support.

# This option exists only if Bochs compiled with --enable-avx option.

#

# FMA4:

# Select AMD four operand FMA instructions support.

# This option exists only if Bochs compiled with --enable-avx option.

#

# TBM:

# Select AMD Trailing Bit Manipulation (TBM) instructions support.

# This option exists only if Bochs compiled with --enable-avx option.

#

# X86-64:

# Enable x86-64 and long mode support.

# This option exists only if Bochs compiled with x86-64 support.

#

# 1G\_PAGES:

# Enable 1G page size support in long mode.

# This option exists only if Bochs compiled with x86-64 support.

#

# PCID:

# Enable Process-Context Identifiers (PCID) support in long mode.

# This option exists only if Bochs compiled with x86-64 support.

#

# FSGSBASE:

# Enable GS/GS BASE access instructions support in long mode.

# This option exists only if Bochs compiled with x86-64 support.

#

# SMEP:

# Enable Supervisor Mode Execution Protection (SMEP) support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# SMAP:

# Enable Supervisor Mode Access Prevention (SMAP) support.

# This option exists only if Bochs compiled with BX\_CPU\_LEVEL >= 6.

#

# MWAIT:

# Select MONITOR/MWAIT instructions support.

# This option exists only if Bochs compiled with --enable-monitor-mwait.

#

# VMX:

# Select VMX extensions emulation support.

# This option exists only if Bochs compiled with --enable-vmx option.

#

# SVM:

# Select AMD SVM (Secure Virtual Machine) extensions emulation support.

# This option exists only if Bochs compiled with --enable-svm option.

#

# VENDOR\_STRING:

# Set the CPUID vendor string returned by CPUID(0x0). This should be a

# twelve-character ASCII string.

#

# BRAND\_STRING:

# Set the CPUID vendor string returned by CPUID(0x80000002 .. 0x80000004).

# This should be at most a forty-eight-character ASCII string.

#

# LEVEL:

# Set emulated CPU level information returned by CPUID. Default value is

# determined by configure option --enable-cpu-level. Currently supported

# values are 5 (for Pentium and similar processors) and 6 (for P6 and

# later processors).

#

# FAMILY:

# Set model information returned by CPUID. Default family value determined

# by configure option --enable-cpu-level.

#

# MODEL:

# Set model information returned by CPUID. Default model value is 3.

#

# STEPPING:

# Set stepping information returned by CPUID. Default stepping value is 3.

#=======================================================================

#cpuid: x86\_64=1, mmx=1, sep=1, simd=sse4\_2, apic=xapic, aes=1, movbe=1, xsave=1

#cpuid: family=6, model=0x1a, stepping=5

#=======================================================================

# MEMORY

# Set the amount of physical memory you want to emulate.

#

# GUEST:

# Set amount of guest physical memory to emulate. The default is 32MB,

# the maximum amount limited only by physical address space limitations.

#

# HOST:

# Set amount of host memory you want to allocate for guest RAM emulation.

# It is possible to allocate less memory than you want to emulate in guest

# system. This will fake guest to see the non-existing memory. Once guest

# system touches new memory block it will be dynamically taken from the

# memory pool. You will be warned (by FATAL PANIC) in case guest already

# used all allocated host memory and wants more.

#

#=======================================================================

memory: guest=512, host=256

#=======================================================================

# ROMIMAGE:

# The ROM BIOS controls what the PC does when it first powers on.

# Normally, you can use a precompiled BIOS in the source or binary

# distribution called BIOS-bochs-latest. The default ROM BIOS is usually loaded

# starting at address 0xfffe0000, and it is exactly 128k long. The legacy

# version of the Bochs BIOS is usually loaded starting at address 0xffff0000,

# and it is exactly 64k long.

# You can use the environment variable $BXSHARE to specify the location

# of the BIOS.

# The usage of external large BIOS images (up to 512k) at memory top is

# now supported, but we still recommend to use the BIOS distributed with Bochs.

# The start address is optional, since it can be calculated from image size.

#=======================================================================

romimage: file=$BXSHARE/BIOS-bochs-latest

#romimage: file=$BXSHARE/bios.bin-1.7.5 # http://www.seabios.org/SeaBIOS

#romimage: file=mybios.bin, address=0xfff80000 # 512k at memory top

#=======================================================================

# VGAROMIMAGE

# You now need to load a VGA ROM BIOS into C0000.

#=======================================================================

#vgaromimage: file=bios/VGABIOS-elpin-2.40

vgaromimage: file=$BXSHARE/VGABIOS-lgpl-latest

#vgaromimage: file=bios/VGABIOS-lgpl-latest-cirrus

#=======================================================================

# OPTROMIMAGE[1-4]:

# You may now load up to 4 optional ROM images. Be sure to use a

# read-only area, typically between C8000 and EFFFF. These optional

# ROM images should not overwrite the rombios (located at

# F0000-FFFFF) and the videobios (located at C0000-C7FFF).

# Those ROM images will be initialized by the bios if they contain

# the right signature (0x55AA) and a valid checksum.

# It can also be a convenient way to upload some arbitrary code/data

# in the simulation, that can be retrieved by the boot loader

#=======================================================================

#optromimage1: file=optionalrom.bin, address=0xd0000

#optromimage2: file=optionalrom.bin, address=0xd1000

#optromimage3: file=optionalrom.bin, address=0xd2000

#optromimage4: file=optionalrom.bin, address=0xd3000

#optramimage1: file=/path/file1.img, address=0x0010000

#optramimage2: file=/path/file2.img, address=0x0020000

#optramimage3: file=/path/file3.img, address=0x0030000

#optramimage4: file=/path/file4.img, address=0x0040000

#=======================================================================

# VGA:

# This defines parameters related to the VGA display

#

# EXTENSION

# Here you can specify the display extension to be used. With the value

# 'none' you can use standard VGA with no extension. Other supported

# values are 'vbe' for Bochs VBE and 'cirrus' for Cirrus SVGA support.

#

# UPDATE\_FREQ

# This parameter specifies the number of display updates per second.

# The VGA update timer now uses the realtime engine and the default

# value is 5. This parameter can be changed at runtime.

#

# REALTIME

# If set to 1, the VGA timer is based on realtime, otherwise it is based

# on the ips setting. If the host is slow (low ips, update\_freq) and the

# guest uses HLT appropriately, setting this to 0 and "clock: sync=none"

# may improve the responsiveness of the guest GUI when the guest is

# otherwise idle. The default value is 1.

#

# Examples:

# vga: extension=cirrus, update\_freq=10

#=======================================================================

#vga: extension=vbe, update\_freq=5, realtime=1

#=======================================================================

# VOODOO:

# This defines the Voodoo Graphics emulation (experimental). Currently

# supported models are 'voodoo1' and 'voodoo2'. The Voodoo2 support is

# not yet complete.

#

# Examples:

# voodoo: enabled=1, model=voodoo1

#=======================================================================

#voodoo: enabled=1, model=voodoo1

#=======================================================================

# KEYBOARD:

# This defines parameters related to the emulated keyboard

#

# TYPE:

# Type of keyboard return by a "identify keyboard" command to the

# keyboard controller. It must be one of "xt", "at" or "mf".

# Defaults to "mf". It should be ok for almost everybody. A known

# exception is french macs, that do have a "at"-like keyboard.

#

# SERIAL\_DELAY:

# Approximate time in microseconds that it takes one character to

# be transferred from the keyboard to controller over the serial path.

#

# PASTE\_DELAY:

# Approximate time in microseconds between attempts to paste

# characters to the keyboard controller. This leaves time for the

# guest os to deal with the flow of characters. The ideal setting

# depends on how your operating system processes characters. The

# default of 100000 usec (.1 seconds) was chosen because it works

# consistently in Windows.

# If your OS is losing characters during a paste, increase the paste

# delay until it stops losing characters.

#

# KEYMAP:

# This enables a remap of a physical localized keyboard to a

# virtualized us keyboard, as the PC architecture expects.

#

# USER\_SHORTCUT:

# This defines the keyboard shortcut to be sent when you press the "user"

# button in the headerbar. The shortcut string is a combination of maximum

# 3 key names (listed below) separated with a '-' character.

# Valid key names:

# "alt", "bksl", "bksp", "ctrl", "del", "down", "end", "enter", "esc",

# "f1", ... "f12", "home", "ins", "left", "menu", "minus", "pgdwn", "pgup",

# "plus", "power", "print", "right", "scrlck", "shift", "space", "tab", "up"

# and "win".

# Examples:

# keyboard: type=mf, serial\_delay=200, paste\_delay=100000

# keyboard: keymap=gui/keymaps/x11-pc-de.map

# keyboard: user\_shortcut=ctrl-alt-del

#=======================================================================

#keyboard: type=mf, serial\_delay=250

#=======================================================================

# MOUSE:

# This defines parameters for the emulated mouse type, the initial status

# of the mouse capture and the runtime method to toggle it.

#

# TYPE:

# With the mouse type option you can select the type of mouse to emulate.

# The default value is 'ps2'. The other choices are 'imps2' (wheel mouse

# on PS/2), 'serial', 'serial\_wheel', 'serial\_msys' (one com port requires

# setting 'mode=mouse') and 'bus' (if present). To connect a mouse to an

# USB port, see the 'usb\_uhci', 'usb\_ohci' or 'usb\_xhci' options (requires

# PCI and USB support).

#

# ENABLED:

# The Bochs gui creates mouse "events" unless the 'enabled' option is

# set to 0. The hardware emulation itself is not disabled by this.

# Unless you have a particular reason for enabling the mouse by default,

# it is recommended that you leave it off. You can also toggle the mouse

# usage at runtime (RFB, SDL, Win32, wxWidgets and X11 - see below).

#

# TOGGLE:

# The default method to toggle the mouse capture at runtime is to press the

# CTRL key and the middle mouse button ('ctrl+mbutton'). This option allows

# to change the method to 'ctrl+f10' (like DOSBox), 'ctrl+alt' (like QEMU)

# or 'f12' (replaces win32 'legacyF12' option).

#

# Examples:

# mouse: enabled=1

# mouse: type=imps2, enabled=1

# mouse: type=serial, enabled=1

# mouse: enabled=0, toggle=ctrl+f10

#=======================================================================

mouse: enabled=0

#=======================================================================

# PCI:

# This option controls the presence of a PCI chipset in Bochs. Currently it only

# supports the i430FX and i440FX chipsets. You can also specify the devices

# connected to PCI slots. Up to 5 slots are available. For these combined PCI/ISA

# devices assigning to slot is mandatory if you want to emulate the PCI model:

# cirrus, ne2k and pcivga. These PCI-only devices are also supported, but they

# are auto-assigned if you don't use the slot configuration: e1000, es1370,

# pcidev, pcipnic, usb\_ohci, usb\_xhci and voodoo.

#

# Example:

# pci: enabled=1, chipset=i440fx, slot1=pcivga, slot2=ne2k

#=======================================================================

pci: enabled=1, chipset=i440fx

#=======================================================================

# CLOCK:

# This defines the parameters of the clock inside Bochs:

#

# SYNC:

# This defines the method how to synchronize the Bochs internal time

# with realtime. With the value 'none' the Bochs time relies on the IPS

# value and no host time synchronization is used. The 'slowdown' method

# sacrifices performance to preserve reproducibility while allowing host

# time correlation. The 'realtime' method sacrifices reproducibility to

# preserve performance and host-time correlation.

# It is possible to enable both synchronization methods.

#

# RTC\_SYNC:

# If this option is enabled together with the realtime synchronization,

# the RTC runs at realtime speed. This feature is disabled by default.

#

# TIME0:

# Specifies the start (boot) time of the virtual machine. Use a time

# value as returned by the time(2) system call or a string as returned

# by the ctime(3) system call. If no time0 value is set or if time0

# equal to 1 (special case) or if time0 equal 'local', the simulation

# will be started at the current local host time. If time0 equal to 2

# (special case) or if time0 equal 'utc', the simulation will be started

# at the current utc time.

#

# Syntax:

# clock: sync=[none|slowdown|realtime|both], time0=[timeValue|local|utc]

#

# Example:

# clock: sync=none, time0=local # Now (localtime)

# clock: sync=slowdown, time0=315529200 # Tue Jan 1 00:00:00 1980

# clock: sync=none, time0="Mon Jan 1 00:00:00 1990" # 631148400

# clock: sync=realtime, time0=938581955 # Wed Sep 29 07:12:35 1999

# clock: sync=realtime, time0="Sat Jan 1 00:00:00 2000" # 946681200

# clock: sync=none, time0=1 # Now (localtime)

# clock: sync=none, time0=utc # Now (utc/gmt)

#

# Default value are sync=none, rtc\_sync=0, time0=local

#=======================================================================

#clock: sync=none, time0=local

#=======================================================================

# CMOSIMAGE:

# This defines image file that can be loaded into the CMOS RAM at startup.

# The rtc\_init parameter controls whether initialize the RTC with values stored

# in the image. By default the time0 argument given to the clock option is used.

# With 'rtc\_init=image' the image is the source for the initial time.

#

# Example:

# cmosimage: file=cmos.img, rtc\_init=image

#=======================================================================

#cmosimage: file=cmos.img, rtc\_init=time0

#=======================================================================

# private\_colormap: Request that the GUI create and use it's own

# non-shared colormap. This colormap will be used

# when in the bochs window. If not enabled, a

# shared colormap scheme may be used. Not implemented

# on all GUI's.

#

# Examples:

# private\_colormap: enabled=1

# private\_colormap: enabled=0

#=======================================================================

private\_colormap: enabled=0

#=======================================================================

# FLOPPYA:

# Point this to pathname of floppy image file or device

# This should be of a bootable floppy(image/device) if you're

# booting from 'a' (or 'floppy').

#

# You can set the initial status of the media to 'ejected' or 'inserted'.

# floppya: 2\_88=path, status=ejected (2.88M 3.5" media)

# floppya: 1\_44=path, status=inserted (1.44M 3.5" media)

# floppya: 1\_2=path, status=ejected (1.2M 5.25" media)

# floppya: 720k=path, status=inserted (720K 3.5" media)

# floppya: 360k=path, status=inserted (360K 5.25" media)

# floppya: 320k=path, status=inserted (320K 5.25" media)

# floppya: 180k=path, status=inserted (180K 5.25" media)

# floppya: 160k=path, status=inserted (160K 5.25" media)

# floppya: image=path, status=inserted (guess media type from image size)

# floppya: 1\_44=vvfat:path, status=inserted (use directory as VFAT media)

# floppya: type=1\_44 (1.44M 3.5" floppy drive, no media)

#

# The path should be the name of a disk image file. On Unix, you can use a raw

# device name such as /dev/fd0 on Linux. On win32 platforms, use drive letters

# such as a: or b: as the path. The parameter 'image' works with image files

# only. In that case the size must match one of the supported types.

# The parameter 'type' can be used to enable the floppy drive without media

# and status specified. Usually the drive type is set up based on the media type.

# The optional parameter 'write\_protected' can be used to control the media

# write protect switch. By default it is turned off.

#=======================================================================

floppya: 1\_44=/dev/fd0, status=inserted

#floppya: image=../1.44, status=inserted

#floppya: 1\_44=/dev/fd0H1440, status=inserted

#floppya: 1\_2=../1\_2, status=inserted

#floppya: 1\_44=a:, status=inserted

#floppya: 1\_44=a.img, status=inserted, write\_protected=1

#floppya: 1\_44=/dev/rfd0a, status=inserted

#=======================================================================

# FLOPPYB:

# See FLOPPYA above for syntax

#=======================================================================

#floppyb: 1\_44=b:, status=inserted

#floppyb: 1\_44=b.img, status=inserted

#=======================================================================

# ATA0, ATA1, ATA2, ATA3

# ATA controller for hard disks and cdroms

#

# ata[0-3]: enabled=[0|1], ioaddr1=addr, ioaddr2=addr, irq=number

#

# These options enables up to 4 ata channels. For each channel

# the two base io addresses and the irq must be specified.

#

# ata0 and ata1 are enabled by default with the values shown below

#

# Examples:

# ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14

# ata1: enabled=1, ioaddr1=0x170, ioaddr2=0x370, irq=15

# ata2: enabled=1, ioaddr1=0x1e8, ioaddr2=0x3e0, irq=11

# ata3: enabled=1, ioaddr1=0x168, ioaddr2=0x360, irq=9

#=======================================================================

ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14

ata1: enabled=1, ioaddr1=0x170, ioaddr2=0x370, irq=15

ata2: enabled=0, ioaddr1=0x1e8, ioaddr2=0x3e0, irq=11

ata3: enabled=0, ioaddr1=0x168, ioaddr2=0x360, irq=9

#=======================================================================

# ATA[0-3]-MASTER, ATA[0-3]-SLAVE

#

# This defines the type and characteristics of all attached ata devices:

# type= type of attached device [disk|cdrom]

# mode= only valid for disks [flat|concat|external|dll|sparse|vmware3]

# [vmware4|undoable|growing|volatile|vpc]

# [vbox|vvfat]

# path= path of the image / directory

# cylinders= only valid for disks

# heads= only valid for disks

# spt= only valid for disks

# status= only valid for cdroms [inserted|ejected]

# biosdetect= type of biosdetection [none|auto], only for disks on ata0 [cmos]

# translation=type of translation of the bios, only for disks [none|lba|large|rechs|auto]

# model= string returned by identify device command

# journal= optional filename of the redolog for undoable, volatile and vvfat disks

#

# Point this at a hard disk image file, cdrom iso file, or physical cdrom

# device. To create a hard disk image, try running bximage. It will help you

# choose the size and then suggest a line that works with it.

#

# In UNIX it may be possible to use a raw device as a Bochs hard disk,

# but WE DON'T RECOMMEND IT. In Windows there is no easy way.

#

# In windows, the drive letter + colon notation should be used for cdroms.

# Depending on versions of windows and drivers, you may only be able to

# access the "first" cdrom in the system. On MacOSX, use path="drive"

# to access the physical drive.

#

# The path is mandatory for hard disks. Disk geometry autodetection works with

# images created by bximage if CHS is set to 0/0/0 (cylinders are calculated

# using heads=16 and spt=63). For other hard disk images and modes the

# cylinders, heads, and spt are mandatory. In all cases the disk size reported

# from the image must be exactly C\*H\*S\*512.

#

# Default values are:

# mode=flat, biosdetect=auto, translation=auto, model="Generic 1234"

#

# The biosdetect option has currently no effect on the bios

#

# Examples:

# ata0-master: type=disk, mode=flat, path=10M.sample, cylinders=306, heads=4, spt=17

# ata0-slave: type=disk, mode=flat, path=20M.sample, cylinders=615, heads=4, spt=17

# ata1-master: type=disk, mode=flat, path=30M.sample, cylinders=615, heads=6, spt=17

# ata1-slave: type=disk, mode=flat, path=46M.sample, cylinders=940, heads=6, spt=17

# ata2-master: type=disk, mode=flat, path=62M.sample, cylinders=940, heads=8, spt=17

# ata2-slave: type=disk, mode=flat, path=112M.sample, cylinders=900, heads=15, spt=17

# ata3-master: type=disk, mode=flat, path=483M.sample, cylinders=1024, heads=15, spt=63

# ata3-slave: type=cdrom, path=iso.sample, status=inserted

#=======================================================================

ata0-master: type=disk, mode=flat, path="30M.sample"

#ata0-master: type=disk, mode=flat, path="30M.sample", cylinders=615, heads=6, spt=17

#ata0-master: type=disk, mode=flat, path="c.img", cylinders=0 # autodetect

#ata0-slave: type=disk, mode=vvfat, path=/bochs/images/vvfat, journal=vvfat.redolog

#ata0-slave: type=cdrom, path=D:, status=inserted

#ata0-slave: type=cdrom, path=/dev/cdrom, status=inserted

#ata0-slave: type=cdrom, path="drive", status=inserted

#ata0-slave: type=cdrom, path=/dev/rcd0d, status=inserted

#=======================================================================

# BOOT:

# This defines the boot sequence. Now you can specify up to 3 boot drives,

# which can be 'floppy', 'disk', 'cdrom' or 'network' (boot ROM).

# Legacy 'a' and 'c' are also supported.

# Examples:

# boot: floppy

# boot: cdrom, disk

# boot: network, disk

# boot: cdrom, floppy, disk

#=======================================================================

#boot: floppy

boot: disk

#=======================================================================

# FLOPPY\_BOOTSIG\_CHECK: disabled=[0|1]

# Enables or disables the 0xaa55 signature check on boot floppies

# Defaults to disabled=0

# Examples:

# floppy\_bootsig\_check: disabled=0

# floppy\_bootsig\_check: disabled=1

#=======================================================================

floppy\_bootsig\_check: disabled=0

#=======================================================================

# LOG:

# Give the path of the log file you'd like Bochs debug and misc. verbiage

# to be written to. If you don't use this option or set the filename to

# '-' the output is written to the console. If you really don't want it,

# make it "/dev/null" (Unix) or "nul" (win32). :^(

#

# Examples:

# log: ./bochs.out

# log: /dev/tty

#=======================================================================

#log: /dev/null

log: bochsout.txt

#=======================================================================

# LOGPREFIX:

# This handles the format of the string prepended to each log line.

# You may use those special tokens :

# %t : 11 decimal digits timer tick

# %i : 8 hexadecimal digits of cpu current eip (ignored in SMP configuration)

# %e : 1 character event type ('i'nfo, 'd'ebug, 'p'anic, 'e'rror)

# %d : 5 characters string of the device, between brackets

#

# Default : %t%e%d

# Examples:

# logprefix: %t-%e-@%i-%d

# logprefix: %i%e%d

#=======================================================================

#logprefix: %t%e%d

#=======================================================================

# LOG CONTROLS

#

# Bochs has four severity levels for event logging.

# panic: cannot proceed. If you choose to continue after a panic,

# don't be surprised if you get strange behavior or crashes.

# error: something went wrong, but it is probably safe to continue the

# simulation.

# info: interesting or useful messages.

# debug: messages useful only when debugging the code. This may

# spit out thousands per second.

#

# For events of each level, you can choose to exit Bochs ('fatal'), 'report'

# or 'ignore'. On some guis you have the additional choice 'ask'. A gui dialog

# appears asks how to proceed.

#

# It is also possible to specify the 'action' to do for each Bochs facility

# separately (e.g. crash on panics from everything except the cdrom, and only

# report those). See the 'log function' module list in the user documentation.

#

# If you are experiencing many panics, it can be helpful to change

# the panic action to report instead of fatal. However, be aware

# that anything executed after a panic is uncharted territory and can

# cause bochs to become unstable. The panic is a "graceful exit," so

# if you disable it you may get a spectacular disaster instead.

#=======================================================================

panic: action=ask

error: action=report

info: action=report

debug: action=ignore, pci=report # report BX\_DEBUG from module 'pci'

#=======================================================================

# DEBUGGER\_LOG:

# Give the path of the log file you'd like Bochs to log debugger output.

# If you really don't want it, make it /dev/null or '-'. :^(

#

# Examples:

# debugger\_log: ./debugger.out

#=======================================================================

#debugger\_log: /dev/null

#debugger\_log: debugger.out

debugger\_log: -

#=======================================================================

# COM1, COM2, COM3, COM4:

# This defines a serial port (UART type 16550A). In the 'term' mode you can

# specify a device to use as com1. This can be a real serial line, or a pty.

# To use a pty (under X/Unix), create two windows (xterms, usually). One of

# them will run bochs, and the other will act as com1. Find out the tty the com1

# window using the `tty' command, and use that as the `dev' parameter.

# Then do `sleep 1000000' in the com1 window to keep the shell from

# messing with things, and run bochs in the other window. Serial I/O to

# com1 (port 0x3f8) will all go to the other window.

# In socket\* and pipe\* (win32 only) modes Bochs becomes either socket/named pipe

# client or server. In client mode it connects to an already running server (if

# connection fails Bochs treats com port as not connected). In server mode it

# opens socket/named pipe and waits until a client application connects to it

# before starting simulation. This mode is useful for remote debugging (e.g.

# with gdb's "target remote host:port" command or windbg's command line option

# -k com:pipe,port=\\.\pipe\pipename). Socket modes use simple TCP communication,

# pipe modes use duplex byte mode pipes.

# Other serial modes are 'null' (no input/output), 'file' (output to a file

# specified as the 'dev' parameter), 'raw' (use the real serial port - under

# construction for win32), 'mouse' (standard serial mouse - requires

# mouse option setting 'type=serial', 'type=serial\_wheel' or 'type=serial\_msys').

#

# Examples:

# com1: enabled=1, mode=null

# com1: enabled=1, mode=mouse

# com2: enabled=1, mode=file, dev=serial.out

# com3: enabled=1, mode=raw, dev=com1

# com3: enabled=1, mode=socket-client, dev=localhost:8888

# com3: enabled=1, mode=socket-server, dev=localhost:8888

# com4: enabled=1, mode=pipe-client, dev=\\.\pipe\mypipe

# com4: enabled=1, mode=pipe-server, dev=\\.\pipe\mypipe

#=======================================================================

#com1: enabled=1, mode=term, dev=/dev/ttyp9

#=======================================================================

# PARPORT1, PARPORT2:

# This defines a parallel (printer) port. When turned on and an output file is

# defined the emulated printer port sends characters printed by the guest OS

# into the output file. On some platforms a device filename can be used to

# send the data to the real parallel port (e.g. "/dev/lp0" on Linux, "lpt1" on

# win32 platforms).

#

# Examples:

# parport1: enabled=1, file="parport.out"

# parport2: enabled=1, file="/dev/lp0"

# parport1: enabled=0

#=======================================================================

parport1: enabled=1, file="parport.out"

#=======================================================================

# SOUND:

# This defines the lowlevel sound driver(s) for the wave (PCM) input / output

# and the MIDI output feature and (if necessary) the devices to be used.

# It can have several of the following properties.

# All properties are in the format sound: property=value

#

# waveoutdrv:

# This defines the driver to be used for the waveout feature.

# Possible values are 'file' (all wave data sent to file), 'dummy' (no

# output) and the platform-dependant drivers 'alsa', 'oss', 'osx', 'sdl'

# and 'win'.

# waveout:

# This defines the device to be used for wave output (if necessary) or

# the output file for the 'file' driver.

# waveindrv:

# This defines the driver to be used for the wavein feature.

# Possible values are 'dummy' (recording silence) and platform-dependent

# drivers 'alsa', 'oss' and 'win'.

# wavein:

# This defines the device to be used for wave output (if necessary).

# midioutdrv:

# This defines the driver to be used for the MIDI output feature.

# Possible values are 'file' (all MIDI data sent to file), 'dummy' (no

# output) and platform-dependent drivers 'alsa', 'oss', 'osx' and 'win'.

# midiout:

# This defines the device to be used for MIDI output (if necessary).

# driver:

# This defines the driver to be used for all sound features with one

# property. Possible values are 'default' (platform default) and all

# other choices described above. Overriding one or more settings with

# the specific driver parameter is possible.

#

# Example for different drivers:

# sound: waveoutdrv=sdl, waveindrv=alsa, midioutdrv=dummy

#=======================================================================

sound: driver=default, waveout=/dev/dsp. wavein=, midiout=

#=======================================================================

# SPEAKER:

# This defines the PC speaker output mode. In the 'sound' mode the beep

# is generated by the square wave generator which is a part of the

# lowlevel sound support. The 'system' mode is only available on Linux

# and Windows. On Linux /dev/console is used for output and on Windows

# the Beep() function. The 'gui' mode forwards the beep to the related

# gui methods (currently only used by the Carbon gui).

#=======================================================================

speaker: enabled=1, mode=sound

#=======================================================================

# SB16:

# This defines the SB16 sound emulation. It can have several of the

# following properties.

# All properties are in the format sb16: property=value

#

# enabled:

# This optional property controls the presence of the SB16 emulation.

# The emulation is turned on unless this property is used and set to 0.

# midimode: This parameter specifies what to do with the MIDI output.

# 0 = no output

# 1 = output to device specified with the sound option (system dependent)

# 2 = MIDI or raw data output to file (depends on file name extension)

# 3 = dual output (mode 1 and 2 at the same time)

# midifile: This is the file where the midi output is stored (midimode 2 or 3).

# wavemode: This parameter specifies what to do with the PCM output.

# 0 = no output

# 1 = output to device specified with the sound option (system dependent)

# 2 = VOC, WAV or raw data output to file (depends on file name extension)

# 3 = dual output (mode 1 and 2 at the same time)

# wavefile: This is the file where the wave output is stored (wavemode 2 or 3).

# loglevel:

# 0=no log

# 1=resource changes, midi program and bank changes

# 2=severe errors

# 3=all errors

# 4=all errors plus all port accesses

# 5=all errors and port accesses plus a lot of extra info

# log: The file to write the sb16 emulator messages to.

# dmatimer:

# microseconds per second for a DMA cycle. Make it smaller to fix

# non-continuous sound. 750000 is usually a good value. This needs a

# reasonably correct setting for the IPS parameter of the CPU option.

#

# Examples for output modes:

# sb16: midimode=2, midifile="output.mid", wavemode=1 # MIDI to file

# sb16: midimode=1, wavemode=3, wavefile="output.wav" # wave to file and device

#=======================================================================

#sb16: midimode=1, wavemode=1, loglevel=2, log=sb16.log, dmatimer=600000

#=======================================================================

# ES1370:

# This defines the ES1370 sound emulation (recording and playback - except

# DAC1+DAC2 output at the same time). The parameter 'enabled' controls the

# presence of the device. The wave and MIDI output can be sent to device, file

# or both using the parameters 'wavemode', 'wavefile', 'midimode' and

# 'midifile'. See the description of these parameters at the SB16 directive.

#

# Examples:

# es1370: enabled=1, wavemode=1 # use 'sound' parameters

# es1370: enabled=1, wavemode=2, wavefile=output.voc # send output to file

#=======================================================================

#es1370: enabled=1, wavemode=1

#=======================================================================

# ne2k: NE2000 compatible ethernet adapter

#

# Format:

# ne2k: enabled=1, ioaddr=IOADDR, irq=IRQ, mac=MACADDR, ethmod=MODULE,

# ethdev=DEVICE, script=SCRIPT, bootrom=BOOTROM

#

# IOADDR, IRQ: You probably won't need to change ioaddr and irq, unless there

# are IRQ conflicts. These arguments are ignored when assign the ne2k to a

# PCI slot.

#

# MAC: The MAC address MUST NOT match the address of any machine on the net.

# Also, the first byte must be an even number (bit 0 set means a multicast

# address), and you cannot use ff:ff:ff:ff:ff:ff because that's the broadcast

# address. For the ethertap module, you must use fe:fd:00:00:00:01. There may

# be other restrictions too. To be safe, just use the b0:c4... address.

#

# ETHDEV: The ethdev value is the name of the network interface on your host

# platform. On UNIX machines, you can get the name by running ifconfig. On

# Windows machines, you must run niclist to get the name of the ethdev.

# Niclist source code is in misc/niclist.c and it is included in Windows

# binary releases.

#

# SCRIPT: The script value is optional, and is the name of a script that

# is executed after bochs initialize the network interface. You can use

# this script to configure this network interface, or enable masquerading.

# This is mainly useful for the tun/tap devices that only exist during

# Bochs execution. The network interface name is supplied to the script

# as first parameter. The 'slirp' module uses this parameter to specify

# a config file for setting up an alternative IP configuration or additional

# features.

#

# BOOTROM: The bootrom value is optional, and is the name of the ROM image

# to load. Note that this feature is only implemented for the PCI version of

# the NE2000.

#

# If you don't want to make connections to any physical networks,

# you can use the following 'ethmod's to simulate a virtual network.

# null: All packets are discarded, but logged to a few files.

# vde: Virtual Distributed Ethernet

# vnet: ARP, ICMP-echo(ping), DHCP and read/write TFTP are simulated.

# The virtual host uses 192.168.10.1.

# DHCP assigns 192.168.10.2 to the guest.

# TFTP uses the 'ethdev' value for the root directory and doesn't

# overwrite files.

#

#=======================================================================

# ne2k: ioaddr=0x300, irq=9, mac=fe:fd:00:00:00:01, ethmod=fbsd, ethdev=en0 #macosx

# ne2k: ioaddr=0x300, irq=9, mac=b0:c4:20:00:00:00, ethmod=fbsd, ethdev=xl0

# ne2k: ioaddr=0x300, irq=9, mac=b0:c4:20:00:00:00, ethmod=linux, ethdev=eth0

# ne2k: ioaddr=0x300, irq=9, mac=b0:c4:20:00:00:01, ethmod=win32, ethdev=MYCARD

# ne2k: ioaddr=0x300, irq=9, mac=fe:fd:00:00:00:01, ethmod=tap, ethdev=tap0

# ne2k: ioaddr=0x300, irq=9, mac=fe:fd:00:00:00:01, ethmod=tuntap, ethdev=/dev/net/tun0, script=./tunconfig

# ne2k: ioaddr=0x300, irq=9, mac=b0:c4:20:00:00:01, ethmod=null, ethdev=eth0

# ne2k: ioaddr=0x300, irq=9, mac=b0:c4:20:00:00:01, ethmod=vde, ethdev="/tmp/vde.ctl"

# ne2k: ioaddr=0x300, irq=9, mac=b0:c4:20:00:00:01, ethmod=vnet, ethdev="c:/temp"

# ne2k: mac=b0:c4:20:00:00:01, ethmod=slirp, script=slirp.conf, bootrom=ne2k\_pci.rom

#=======================================================================

# pcipnic: Bochs/Etherboot pseudo-NIC

#

# Format:

# pcipnic: enabled=1, mac=MACADDR, ethmod=MODULE, ethdev=DEVICE, script=SCRIPT,

# bootrom=BOOTROM

#

# The pseudo-NIC accepts the same syntax (for mac, ethmod, ethdev, script,

# bootrom) and supports the same networking modules as the NE2000 adapter.

#=======================================================================

#pcipnic: enabled=1, mac=b0:c4:20:00:00:00, ethmod=vnet

#=======================================================================

# e1000: Intel(R) 82540EM Gigabit Ethernet adapter

#

# Format:

# e1000: enabled=1, mac=MACADDR, ethmod=MODULE, ethdev=DEVICE, script=SCRIPT

# bootrom=BOOTROM

#

# The E1000 accepts the same syntax (for mac, ethmod, ethdev, script, bootrom)

# and supports the same networking modules as the NE2000 adapter.

#=======================================================================

#e1000: enabled=1, mac=52:54:00:12:34:56, ethmod=slirp, script=slirp.conf

#=======================================================================

# USB\_UHCI:

# This option controls the presence of the USB root hub which is a part

# of the i440FX PCI chipset. With the portX parameter you can connect devices

# to the hub (currently supported: 'mouse', 'tablet', 'keypad', 'disk', 'cdrom'

# 'hub' and 'printer').

#

# If you connect the mouse or tablet to one of the ports, Bochs forwards the

# mouse movement data to the USB device instead of the selected mouse type.

# When connecting the keypad to one of the ports, Bochs forwards the input of

# the numeric keypad to the USB device instead of the PS/2 keyboard.

#

# To connect a 'flat' mode image as an USB hardisk you can use the 'disk' device

# with the path to the image separated with a colon. To use other disk image modes

# similar to ATA disks the syntax 'disk:mode:filename' must be used (see below).

#

# To emulate an USB cdrom you can use the 'cdrom' device name and the path to

# an ISO image or raw device name also separated with a colon. An option to

# insert/eject media is available in the runtime configuration.

#

# The device name 'hub' connects an external hub with max. 8 ports (default: 4)

# to the root hub. To specify the number of ports you have to add the value

# separated with a colon. Connecting devices to the external hub ports is only

# available in the runtime configuration.

#

# The device 'printer' emulates the HP Deskjet 920C printer. The PCL data is

# sent to a file specified in bochsrc.txt. The current code appends the PCL

# code to the file if the file already existed. It would probably be nice to

# overwrite the file instead, asking user first.

#

# The optionsX parameter can be used to assign specific options to the device

# connected to the corresponding USB port. Currently this feature is used to

# set the speed reported by device ('low', 'full', 'high' or 'super'). The

# availabe speed choices depend on both HC and device. For the USB 'disk' device

# the optionsX parameter can be used to specify an alternative redolog file

# (journal) of some image modes. For 'vvfat' mode USB disks the optionsX

# parameter can be used to specify the disk size (range 128M ... 128G). If the

# size is not specified, it defaults to 504M.

#=======================================================================

#usb\_uhci: enabled=1

#usb\_uhci: enabled=1, port1=mouse, port2=disk:usbstick.img

#usb\_uhci: enabled=1, port1=hub:7, port2=disk:growing:usbdisk.img

#usb\_uhci: enabled=1, port2=disk:undoable:usbdisk.img, options2=journal:redo.log

#usb\_uhci: enabled=1, port2=disk:vvfat:vvfat, options2=speed:full

#usb\_uhci: enabled=1, port1=printer:printdata.bin, port2=cdrom:image.iso

#=======================================================================

# USB\_OHCI:

# This option controls the presence of the USB OHCI host controller with a

# 2-port hub. The portX parameter accepts the same device types with the same

# syntax as the UHCI controller (see above). The optionsX parameter is also

# available on OHCI.

#=======================================================================

#usb\_ohci: enabled=1

#usb\_ohci: enabled=1, port1=printer:usbprinter.bin

#=======================================================================

# USB\_XHCI:

# This option controls the presence of the experimental USB xHCI host controller

# with a 4-port hub. The portX parameter accepts the same device types with the

# same syntax as the UHCI controller (see above). The optionsX parameter is

# also available on xHCI. NOTE: port 1 and 2 are USB3 and only support

# super-speed devices, but port 3 and 4 are USB2 and support speed settings

# low, full and high.

#=======================================================================

#usb\_xhci: enabled=1

#=======================================================================

# PCIDEV:

# PCI host device mapping

#=======================================================================

#pcidev: vendor=0x1234, device=0x5678

#=======================================================================

# GDBSTUB:

# Enable GDB stub. See user documentation for details.

# Default value is enabled=0.

#=======================================================================

#gdbstub: enabled=0, port=1234, text\_base=0, data\_base=0, bss\_base=0

#=======================================================================

# MAGIC\_BREAK:

# This enables the "magic breakpoint" feature when using the debugger.

# The useless cpu instruction XCHG BX, BX causes Bochs to enter the

# debugger mode. This might be useful for software development.

#

# Example:

# magic\_break: enabled=1

#=======================================================================

#magic\_break: enabled=1

#=======================================================================

# DEBUG\_SYMBOLS:

# This loads symbols from the specified file for use in Bochs' internal

# debugger. Symbols are loaded into global context. This is equivalent to

# issuing ldsym debugger command at start up.

#

# Example:

# debug\_symbols: file="kernel.sym"

# debug\_symbols: file="kernel.sym", offset=0x80000000

#=======================================================================

#debug\_symbols: file="kernel.sym"

#print\_timestamps: enabled=1

#=======================================================================

# PORT\_E9\_HACK:

# The 0xE9 port doesn't exists in normal ISA architecture. However, we

# define a convention here, to display on the console of the system running

# Bochs anything that is written to it. The idea is to provide debug output

# very early when writing BIOS or OS code for example, without having to

# bother with setting up a serial port or etc. Reading from port 0xE9 will

# will return 0xe9 to let you know if the feature is available.

# Leave this 0 unless you have a reason to use it.

#

# Example:

# port\_e9\_hack: enabled=1

#=======================================================================

#port\_e9\_hack: enabled=1

#=======================================================================

# other stuff

#=======================================================================

#load32bitOSImage: os=nullkernel, path=../kernel.img, iolog=../vga\_io.log

#load32bitOSImage: os=linux, path=../linux.img, iolog=../vga\_io.log, initrd=../initrd.img

#=======================================================================

# fullscreen: ONLY IMPLEMENTED ON AMIGA

# Request that Bochs occupy the entire screen instead of a

# window.

#

# Examples:

# fullscreen: enabled=0

# fullscreen: enabled=1

#=======================================================================

#fullscreen: enabled=0

#screenmode: name="sample"

#=======================================================================

# USER\_PLUGIN:

# Load user-defined plugin. This option is available only if Bochs is

# compiled with plugin support. Maximum 8 different plugins are supported.

# See the example in the Bochs sources how to write a plugin device.

#=======================================================================

#user\_plugin: name=testdev

#=======================================================================

# for Macintosh, use the style of pathnames in the following

# examples.

#

# vgaromimage: :bios:VGABIOS-elpin-2.40

# romimage: file=:bios:BIOS-bochs-latest, address=0xf0000

# floppya: 1\_44=[fd:], status=inserted

#=======================================================================

#=======================================================================

# MEGS

# Set the number of Megabytes of physical memory you want to emulate.

# The default is 32MB, most OS's won't need more than that.

# The maximum amount of memory supported is 2048Mb.

# The 'MEGS' option is deprecated. Use 'MEMORY' option instead.

#=======================================================================

#megs: 256

#megs: 128

#megs: 64

#megs: 32

#megs: 16

#megs: 8