```
#! /bin/sh
#
#  Turtle Beach MultiSound Driver Notes
#  -- Andrew Veliath <andrewtv@usa.net>
#
#  Last update:                      September 10, 1998
#  Corresponding msnd driver:        0.8.3
#
# ** This file is a README (top part) and shell archive (bottom part).
#    The corresponding archived utility sources can be unpacked by
#    running `sh MultiSound' (the utilities are only needed for the
#    Pinnacle and Fiji cards). **
#
#
#  -=-=- Getting Firmware -=-=-
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#  See the section `Obtaining and Creating Firmware Files' in this
#  document for instructions on obtaining the necessary firmware
#  files.
#
#
#  Supported Features
#  ~~~~~~~~~~~~~~~~~~
#
#  Currently, full-duplex digital audio (/dev/dsp only, /dev/audio is
#  not currently available) and mixer functionality (/dev/mixer) are
#  supported (memory mapped digital audio is not yet supported).
#  Digital transfers and monitoring can be done as well if you have
#  the digital daughterboard (see the section on using the S/PDIF port
#  for more information).
#
#  Support for the Turtle Beach MultiSound Hurricane architecture is
#  composed of the following modules (these can also operate compiled
#  into the kernel):
#
#  msnd               - MultiSound base (requires soundcore)
#
#  msnd_classic       - Base audio/mixer support for Classic, Monetery and
#                       Tahiti cards
#
#  msnd_pinnacle      - Base audio/mixer support for Pinnacle and Fiji cards
#
#
#  Important Notes - Read Before Using
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#  The firmware files are not included (may change in future).  You
#  must obtain these images from Turtle Beach (they are included in
#  the MultiSound Development Kits), and place them in /etc/sound for
#  example, and give the full paths in the Linux configuration.  If
#  you are compiling in support for the MultiSound driver rather than
#  using it as a module, these firmware files must be accessible
#  during kernel compilation.
#
#  Please note these files must be binary files, not assembler.  See
```

```
#  the section later in this document for instructions to obtain these
#  files.
#
#
#  Configuring Card Resources
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#  ** This section is very important, as your card may not work at all
#     or your machine may crash if you do not do this correctly. **
#
#  * Classic/Monterey/Tahiti
#
#  These cards are configured through the driver msnd_classic.  You must
#  know the io port, then the driver will select the irq and memory resources
#  on the card.  It is up to you to know if these are free locations or now,
#  a conflict can lock the machine up.
#
#  * Pinnacle/Fiji
#
#  The Pinnacle and Fiji cards have an extra config port, either
#  0x250, 0x260 or 0x270.  This port can be disabled to have the card
#  configured strictly through PnP, however you lose the ability to
#  access the IDE controller and joystick devices on this card when
#  using PnP.  The included pinnaclecfg program in this shell archive
#  can be used to configure the card in non-PnP mode, and in PnP mode
#  you can use isapnptools.  These are described briefly here.
#
#  pinnaclecfg is not required; you can use the msnd_pinnacle module
#  to fully configure the card as well.  However, pinnaclecfg can be
#  used to change the resource values of a particular device after the
#  msnd_pinnacle module has been loaded.  If you are compiling the
#  driver into the kernel, you must set these values during compile
#  time, however other peripheral resource values can be changed with
#  the pinnaclecfg program after the kernel is loaded.
#
#
#  *** PnP mode
#
#  Use pnpdump to obtain a sample configuration if you can; I was able
#  to obtain one with the command `pnpdump 1 0x203' -- this may vary
#  for you (running pnpdump by itself did not work for me).  Then,
#  edit this file and use isapnp to uncomment and set the card values.
#  Use these values when inserting the msnd_pinnacle module.  Using
#  this method, you can set the resources for the DSP and the Kurzweil
#  synth (Pinnacle).  Since Linux does not directly support PnP
#  devices, you may have difficulty when using the card in PnP mode
#  when it the driver is compiled into the kernel.  Using non-PnP mode
#  is preferable in this case.
#
#  Here is an example mypinnacle.conf for isapnp that sets the card to
#  io base 0x210, irq 5 and mem 0xd8000, and also sets the Kurzweil
#  synth to 0x330 and irq 9 (may need editing for your system):
#
#  (READPORT 0x0203)
#  (CSN 2)
#  (IDENTIFY *)
```

```
#
#  # DSP
#  (CONFIGURE BVJ0440/-1 (LD 0
#          (INT 0 (IRQ 5 (MODE +E))) (IO 0 (BASE 0x0210)) (MEM 0 (BASE
0x0d8000))
#          (ACT Y)))
#
#  # Kurzweil Synth (Pinnacle Only)
#  (CONFIGURE BVJ0440/-1 (LD 1
#          (IO 0 (BASE 0x0330)) (INT 0 (IRQ 9 (MODE +E)))
#          (ACT Y)))
#
#  (WAITFORKEY)
#
#
#  *** Non-PnP mode
#
#  The second way is by running the card in non-PnP mode.  This
#  actually has some advantages in that you can access some other
#  devices on the card, such as the joystick and IDE controller.  To
#  configure the card, unpack this shell archive and build the
#  pinnaclecfg program.  Using this program, you can assign the
#  resource values to the card's devices, or disable the devices.  As
#  an alternative to using pinnaclecfg, you can specify many of the
#  configuration values when loading the msnd_pinnacle module (or
#  during kernel configuration when compiling the driver into the
#  kernel).
#
#  If you specify cfg=0x250 for the msnd_pinnacle module, it
#  automatically configure the card to the given io, irq and memory
#  values using that config port (the config port is jumper selectable
#  on the card to 0x250, 0x260 or 0x270).
#
#  See the `msnd_pinnacle Additional Options' section below for more
#  information on these parameters (also, if you compile the driver
#  directly into the kernel, these extra parameters can be useful
#  here).
#
#
# ** It is very easy to cause problems in your machine if you choose a
#    resource value which is incorrect. **
#
#
#  Examples
#  ~~~~~~~~
#
#
#  * MultiSound Classic/Monterey/Tahiti:
#
#  modprobe soundcore
#  insmod msnd
#  insmod msnd_classic io=0x290 irq=7 mem=0xd0000
#
#  * MultiSound Pinnacle in PnP mode:
#
#  modprobe soundcore
#  insmod msnd
```

```
#  isapnp mypinnacle.conf
#  insmod msnd_pinnacle io=0x210 irq=5 mem=0xd8000 <-- match mypinnacle.conf
values
#
#  * MultiSound Pinnacle in non-PnP mode (replace 0x250 with your configuration
port,
#    one of 0x250, 0x260 or 0x270):
#
#  insmod soundcore
#  insmod msnd
#  insmod msnd_pinnacle cfg=0x250 io=0x290 irq=5 mem=0xd0000
#
# * To use the MPU-compatible Kurzweil synth on the Pinnacle in PnP
#   mode, add the following (assumes you did `isapnp mypinnacle.conf'):
#
#  insmod sound
#  insmod mpu401 io=0x330 irq=9                    <-- match mypinnacle.conf
values
#
# * To use the MPU-compatible Kurzweil synth on the Pinnacle in non-PnP
#   mode, add the following.  Note how we first configure the peripheral's
#   resources, _then_ install a Linux driver for it:
#
#  insmod sound
#  pinnaclecfg 0x250 mpu 0x330 9
#  insmod mpu401 io=0x330 irq=9
#
#  -- OR you can use the following sequence without pinnaclecfg in non-PnP mode:
#
#  insmod soundcore
#  insmod msnd
#  insmod msnd_pinnacle cfg=0x250 io=0x290 irq=5 mem=0xd0000 mpu_io=0x330
mpu_irq=9
#  insmod sound
#  insmod mpu401 io=0x330 irq=9
#
# * To setup the joystick port on the Pinnacle in non-PnP mode (though
#   you have to find the actual Linux joystick driver elsewhere), you
#   can use pinnaclecfg:
#
#   pinnaclecfg 0x250 joystick 0x200
#
#  -- OR you can configure this using msnd_pinnacle with the following:
#
#  insmod soundcore
#  insmod msnd
#  insmod msnd_pinnacle cfg=0x250 io=0x290 irq=5 mem=0xd0000 joystick_io=0x200
#
#
#  msnd_classic, msnd_pinnacle Required Options
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#  If the following options are not given, the module will not load.
#  Examine the kernel message log for informative error messages.
#  WARNING--probing isn't supported so try to make sure you have the
#  correct shared memory area, otherwise you may experience problems.
```

```
#
#  io                    I/O base of DSP, e.g. io=0x210
#  irq                   IRQ number, e.g. irq=5
#  mem                   Shared memory area, e.g. mem=0xd8000
#
#
#  msnd_classic, msnd_pinnacle Additional Options
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#
#  fifosize              The digital audio FIFOs, in kilobytes.  If not
#                        specified, the default will be used.  Increasing
#                        this value will reduce the chance of a FIFO
#                        underflow at the expense of increasing overall
#                        latency.  For example, fifosize=512 will
#                        allocate 512kB read and write FIFOs (1MB total).
#                        While this may reduce dropouts, a heavy machine
#                        load will undoubtedly starve the FIFO of data
#                        and you will eventually get dropouts.  One
#                        option is to alter the scheduling priority of
#                        the playback process, using `nice' or some form
#                        of POSIX soft real-time scheduling.
#
#  calibrate_signal      Setting this to one calibrates the ADCs to the
#                        signal, zero calibrates to the card (defaults
#                        to zero).
#
#
#  msnd_pinnacle Additional Options
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#
#  digital               Specify digital=1 to enable the S/PDIF input
#                        if you have the digital daughterboard
#                        adapter. This will enable access to the
#                        DIGITAL1 input for the soundcard in the mixer.
#                        Some mixer programs might have trouble setting
#                        the DIGITAL1 source as an input.  If you have
#                        trouble, you can try the setdigital.c program
#                        at the bottom of this document.
#
#  cfg                   Non-PnP configuration port for the Pinnacle
#                        and Fiji (typically 0x250, 0x260 or 0x270,
#                        depending on the jumper configuration).  If
#                        this option is omitted, then it is assumed
#                        that the card is in PnP mode, and that the
#                        specified DSP resource values are already
#                        configured with PnP (i.e. it won't attempt to
#                        do any sort of configuration).
#
#  When the Pinnacle is in non-PnP mode, you can use the following
#  options to configure particular devices.  If a full specification
#  for a device is not given, then the device is not configured.  Note
#  that you still must use a Linux driver for any of these devices
#  once their resources are setup (such as the Linux joystick driver,
#  or the MPU401 driver from OSS for the Kurzweil synth).
#
#  mpu_io                I/O port of MPU (on-board Kurzweil synth)
```

```
#  mpu_irq              IRQ of MPU (on-board Kurzweil synth)
#  ide_io0              First I/O port of IDE controller
#  ide_io1              Second I/O port of IDE controller
#  ide_irq              IRQ IDE controller
#  joystick_io          I/O port of joystick
#
#
#  Obtaining and Creating Firmware Files
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#
#       For the Classic/Tahiti/Monterey
#       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#
#  Download to /tmp and unzip the following file from Turtle Beach:
#
#       ftp://ftp.voyetra.com/pub/tbs/msndcl/msndvkit.zip
#
#  When unzipped, unzip the file named MsndFiles.zip.  Then copy the
#  following firmware files to /etc/sound (note the file renaming):
#
#    cp DSPCODE/MSNDINIT.BIN /etc/sound/msndinit.bin
#    cp DSPCODE/MSNDPERM.REB /etc/sound/msndperm.bin
#
#  When configuring the Linux kernel, specify /etc/sound/msndinit.bin and
#  /etc/sound/msndperm.bin for the two firmware files (Linux kernel
#  versions older than 2.2 do not ask for firmware paths, and are
#  hardcoded to /etc/sound).
#
#  If you are compiling the driver into the kernel, these files must
#  be accessible during compilation, but will not be needed later.
#  The files must remain, however, if the driver is used as a module.
#
#
#       For the Pinnacle/Fiji
#       ~~~~~~~~~~~~~~~~~~~~~
#
#
#  Download to /tmp and unzip the following file from Turtle Beach (be
#  sure to use the entire URL; some have had trouble navigating to the
#  URL):
#
#       ftp://ftp.voyetra.com/pub/tbs/pinn/pnddk100.zip
#
#  Unpack this shell archive, and run make in the created directory
#  (you need a C compiler and flex to build the utilities).  This
#  should give you the executables conv, pinnaclecfg and setdigital.
#  conv is only used temporarily here to create the firmware files,
#  while pinnaclecfg is used to configure the Pinnacle or Fiji card in
#  non-PnP mode, and setdigital can be used to set the S/PDIF input on
#  the mixer (pinnaclecfg and setdigital should be copied to a
#  convenient place, possibly run during system initialization).
#
#  To generating the firmware files with the `conv' program, we create
#  the binary firmware files by doing the following conversion
#  (assuming the archive unpacked into a directory named PINNDDK):
#
#    ./conv < PINNDDK/dspcode/pndspini.asm > /etc/sound/pndspini.bin
```

```
#    ./conv < PINNDDK/dspcode/pndsperm.asm > /etc/sound/pndsperm.bin
#
#  The conv (and conv.l) program is not needed after conversion and can
#  be safely deleted.  Then, when configuring the Linux kernel, specify
#  /etc/sound/pndspini.bin and /etc/sound/pndsperm.bin for the two
#  firmware files (Linux kernel versions older than 2.2 do not ask for
#  firmware paths, and are hardcoded to /etc/sound).
#
#  If you are compiling the driver into the kernel, these files must
#  be accessible during compilation, but will not be needed later.
#  The files must remain, however, if the driver is used as a module.
#
#
#  Using Digital I/O with the S/PDIF Port
#  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
#  If you have a Pinnacle or Fiji with the digital daughterboard and
#  want to set it as the input source, you can use this program if you
#  have trouble trying to do it with a mixer program (be sure to
#  insert the module with the digital=1 option, or say Y to the option
#  during compiled-in kernel operation).  Upon selection of the S/PDIF
#  port, you should be able monitor and record from it.
#
#  There is something to note about using the S/PDIF port.  Digital
#  timing is taken from the digital signal, so if a signal is not
#  connected to the port and it is selected as recording input, you
#  will find PCM playback to be distorted in playback rate.  Also,
#  attempting to record at a sampling rate other than the DAT rate may
#  be problematic (i.e. trying to record at 8000Hz when the DAT signal
#  is 44100Hz).  If you have a problem with this, set the recording
#  input to analog if you need to record at a rate other than that of
#  the DAT rate.
#
#
#  -- Shell archive attached below, just run `sh MultiSound' to extract.
#     Contains Pinnacle/Fiji utilities to convert firmware, configure
#     in non-PnP mode, and select the DIGITAL1 input for the mixer.
#
#
#!/bin/sh
# This is a shell archive (produced by GNU sharutils 4.2).
# To extract the files from this archive, save it to some FILE, remove
# everything before the `!/bin/sh' line above, then type `sh FILE'.
#
# Made on 1998-12-04 10:07 EST by <andrewtv@ztransform.velsoft.com>.
# Source directory was `/home/andrewtv/programming/pinnacle/pinnacle'.
#
# Existing files will *not* be overwritten unless `-c' is specified.
#
# This shar contains:
# length mode       name
# ------ ---------- ------------------------------------------
#   2046 -rw-rw-r-- MultiSound.d/setdigital.c
#  10235 -rw-rw-r-- MultiSound.d/pinnaclecfg.c
#    106 -rw-rw-r-- MultiSound.d/Makefile
#    141 -rw-rw-r-- MultiSound.d/conv.l
```

```
#   1472 -rw-rw-r-- MultiSound.d/msndreset.c
#
save_IFS="${IFS}"
IFS="${IFS}:"
gettext_dir=FAILED
locale_dir=FAILED
first_param="$1"
for dir in $PATH
do
  if test "$gettext_dir" = FAILED && test -f $dir/gettext \
     && ($dir/gettext --version >/dev/null 2>&1)
  then
    set `$dir/gettext --version 2>&1`
    if test "$3" = GNU
    then
      gettext_dir=$dir
    fi
  fi
  if test "$locale_dir" = FAILED && test -f $dir/shar \
     && ($dir/shar --print-text-domain-dir >/dev/null 2>&1)
  then
    locale_dir=`$dir/shar --print-text-domain-dir`
  fi
done
IFS="$save_IFS"
if test "$locale_dir" = FAILED || test "$gettext_dir" = FAILED
then
  echo=echo
else
  TEXTDOMAINDIR=$locale_dir
  export TEXTDOMAINDIR
  TEXTDOMAIN=sharutils
  export TEXTDOMAIN
  echo="$gettext_dir/gettext -s"
fi
touch -am 1231235999 $$.touch >/dev/null 2>&1
if test ! -f 1231235999 && test -f $$.touch; then
  shar_touch=touch
else
  shar_touch=:
  echo
  $echo 'WARNING: not restoring timestamps.  Consider getting and'
  $echo "installing GNU \`touch', distributed in GNU File Utilities..."
  echo
fi
rm -f 1231235999 $$.touch
#
if mkdir _sh01426; then
  $echo 'x -' 'creating lock directory'
else
  $echo 'failed to create lock directory'
  exit 1
fi
# ============= MultiSound.d/setdigital.c ==============
if test ! -d 'MultiSound.d'; then
  $echo 'x -' 'creating directory' 'MultiSound.d'
  mkdir 'MultiSound.d'
fi
if test -f 'MultiSound.d/setdigital.c' && test "$first_param" != -c; then
  $echo 'x -' SKIPPING 'MultiSound.d/setdigital.c' '(file already exists)'
else
  $echo 'x -' extracting 'MultiSound.d/setdigital.c' '(text)'
  sed 's/^X//' << 'SHAR_EOF' > 'MultiSound.d/setdigital.c' &&
/*********************************************************************
X *
X * setdigital.c - sets the DIGITAL1 input for a mixer
X *
X * Copyright (C) 1998 Andrew Veliath
X *
X * This program is free software; you can redistribute it and/or modify
X * it under the terms of the GNU General Public License as published by
X * the Free Software Foundation; either version 2 of the License, or
X * (at your option) any later version.
X *
X * This program is distributed in the hope that it will be useful,
X * but WITHOUT ANY WARRANTY; without even the implied warranty of
X * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
X * GNU General Public License for more details.
X *
X * You should have received a copy of the GNU General Public License
X * along with this program; if not, write to the Free Software
X * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
X *
X ********************************************************************/
X
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/soundcard.h>
X
int main(int argc, char *argv[])
{
X       int fd;
X       unsigned long recmask, recsrc;
X
X       if (argc != 2) {
X               fprintf(stderr, "usage: setdigital <mixer device>\n");
X               exit(1);
X       }
X
X       if ((fd = open(argv[1], O_RDWR)) < 0) {
X               perror(argv[1]);
X               exit(1);
X       }
X
X       if (ioctl(fd, SOUND_MIXER_READ_RECMASK, &recmask) < 0) {
X               fprintf(stderr, "error: ioctl read recording mask failed\n");
X               perror("ioctl");
X               close(fd);
X               exit(1);
X       }
X
X       if (!(recmask & SOUND_MASK_DIGITAL1)) {
X               fprintf(stderr, "error: cannot find DIGITAL1 device in
mixer\n");
X               close(fd);
X               exit(1);
X       }
X
X       if (ioctl(fd, SOUND_MIXER_READ_RECSRC, &recsrc) < 0) {
X               fprintf(stderr, "error: ioctl read recording source failed\n");
X               perror("ioctl");
X               close(fd);
X               exit(1);
X       }
X
X       recsrc |= SOUND_MASK_DIGITAL1;
X
X       if (ioctl(fd, SOUND_MIXER_WRITE_RECSRC, &recsrc) < 0) {
X               fprintf(stderr, "error: ioctl write recording source failed\n");
X               perror("ioctl");
X               close(fd);
X               exit(1);
X       }
X
X       close(fd);
X
X       return 0;
}
SHAR_EOF
  $shar_touch -am 1204092598 'MultiSound.d/setdigital.c' &&
  chmod 0664 'MultiSound.d/setdigital.c' ||
  $echo 'restore of' 'MultiSound.d/setdigital.c' 'failed'
  if ( md5sum --help 2>&1 | grep 'sage: md5sum \[' ) >/dev/null 2>&1 \
  && ( md5sum --version 2>&1 | grep -v 'textutils 1.12' ) >/dev/null; then
    md5sum -c << SHAR_EOF >/dev/null 2>&1 \
    || $echo 'MultiSound.d/setdigital.c:' 'MD5 check failed'
e87217fc3e71288102ba41fd81f71ec4  MultiSound.d/setdigital.c
SHAR_EOF
  else
    shar_count="`LC_ALL= LC_CTYPE= LANG= wc -c < 'MultiSound.d/setdigital.c'`"
    test 2046 -eq "$shar_count" ||
    $echo 'MultiSound.d/setdigital.c:' 'original size' '2046,' 'current size'
"$shar_count!"
  fi
fi
# ============= MultiSound.d/pinnaclecfg.c ==============
if test -f 'MultiSound.d/pinnaclecfg.c' && test "$first_param" != -c; then
  $echo 'x -' SKIPPING 'MultiSound.d/pinnaclecfg.c' '(file already exists)'
else
  $echo 'x -' extracting 'MultiSound.d/pinnaclecfg.c' '(text)'
  sed 's/^X//' << 'SHAR_EOF' > 'MultiSound.d/pinnaclecfg.c' &&
/*********************************************************************
X *
X * pinnaclecfg.c - Pinnacle/Fiji Device Configuration Program
```

```
X *
X * This is for NON-PnP mode only.  For PnP mode, use isapnptools.
X *
X * This is Linux-specific, and must be run with root permissions.
X *
X * Part of the Turtle Beach MultiSound Sound Card Driver for Linux
X *
X * Copyright (C) 1998 Andrew Veliath
X *
X * This program is free software; you can redistribute it and/or modify
X * it under the terms of the GNU General Public License as published by
X * the Free Software Foundation; either version 2 of the License, or
X * (at your option) any later version.
X *
X * This program is distributed in the hope that it will be useful,
X * but WITHOUT ANY WARRANTY; without even the implied warranty of
X * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
X * GNU General Public License for more details.
X *
X * You should have received a copy of the GNU General Public License
X * along with this program; if not, write to the Free Software
X * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
X *
X ********************************************************************/
X
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <asm/io.h>
#include <asm/types.h>
X
#define IREG_LOGDEVICE          0x07
#define IREG_ACTIVATE           0x30
#define LD_ACTIVATE             0x01
#define LD_DISACTIVATE          0x00
#define IREG_EECONTROL          0x3F
#define IREG_MEMBASEHI          0x40
#define IREG_MEMBASELO          0x41
#define IREG_MEMCONTROL         0x42
#define IREG_MEMRANGEHI         0x43
#define IREG_MEMRANGELO         0x44
#define MEMTYPE_8BIT            0x00
#define MEMTYPE_16BIT           0x02
#define MEMTYPE_RANGE           0x00
#define MEMTYPE_HIADDR          0x01
#define IREG_IO0_BASEHI         0x60
#define IREG_IO0_BASELO         0x61
#define IREG_IO1_BASEHI         0x62
#define IREG_IO1_BASELO         0x63
#define IREG_IRQ_NUMBER         0x70
#define IREG_IRQ_TYPE           0x71
#define IRQTYPE_HIGH            0x02
#define IRQTYPE_LOW             0x00
#define IRQTYPE_LEVEL           0x01
```

```
#define IRQTYPE_EDGE            0x00
X
#define HIBYTE(w)               ((BYTE)(((WORD)(w) >> 8) & 0xFF))
#define LOBYTE(w)               ((BYTE)(w))
#define MAKEWORD(low,hi)
((WORD)(((BYTE)(low))|(((WORD)((BYTE)(hi)))<<8)))
X
typedef __u8                    BYTE;
typedef __u16                   USHORT;
typedef __u16                   WORD;
X
static int config_port = -1;
X
static int msnd_write_cfg(int cfg, int reg, int value)
{
X       outb(reg, cfg);
X       outb(value, cfg + 1);
X       if (value != inb(cfg + 1)) {
X               fprintf(stderr, "error: msnd_write_cfg: I/O error\n");
X               return -EIO;
X       }
X       return 0;
}
X
static int msnd_read_cfg(int cfg, int reg)
{
X       outb(reg, cfg);
X       return inb(cfg + 1);
}
X
static int msnd_write_cfg_io0(int cfg, int num, WORD io)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_IO0_BASEHI, HIBYTE(io)))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_IO0_BASELO, LOBYTE(io)))
X               return -EIO;
X       return 0;
}
X
static int msnd_read_cfg_io0(int cfg, int num, WORD *io)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X
X       *io = MAKEWORD(msnd_read_cfg(cfg, IREG_IO0_BASELO),
X                      msnd_read_cfg(cfg, IREG_IO0_BASEHI));
X
X       return 0;
}
X
static int msnd_write_cfg_io1(int cfg, int num, WORD io)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
```

```
X       if (msnd_write_cfg(cfg, IREG_IO1_BASEHI, HIBYTE(io)))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_IO1_BASELO, LOBYTE(io)))
X               return -EIO;
X       return 0;
}
X
static int msnd_read_cfg_io1(int cfg, int num, WORD *io)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X
X       *io = MAKEWORD(msnd_read_cfg(cfg, IREG_IO1_BASELO),
X                      msnd_read_cfg(cfg, IREG_IO1_BASEHI));
X
X       return 0;
}
X
static int msnd_write_cfg_irq(int cfg, int num, WORD irq)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_IRQ_NUMBER, LOBYTE(irq)))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_IRQ_TYPE, IRQTYPE_EDGE))
X               return -EIO;
X       return 0;
}
X
static int msnd_read_cfg_irq(int cfg, int num, WORD *irq)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X
X       *irq = msnd_read_cfg(cfg, IREG_IRQ_NUMBER);
X
X       return 0;
}
X
static int msnd_write_cfg_mem(int cfg, int num, int mem)
{
X       WORD wmem;
X
X       mem >>= 8;
X       mem &= 0xfff;
X       wmem = (WORD)mem;
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_MEMBASEHI, HIBYTE(wmem)))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_MEMBASELO, LOBYTE(wmem)))
X               return -EIO;
X       if (wmem && msnd_write_cfg(cfg, IREG_MEMCONTROL, (MEMTYPE_HIADDR |
MEMTYPE_16BIT)))
X               return -EIO;
X       return 0;
```

```
}
X
static int msnd_read_cfg_mem(int cfg, int num, int *mem)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X
X       *mem = MAKEWORD(msnd_read_cfg(cfg, IREG_MEMBASELO),
X                       msnd_read_cfg(cfg, IREG_MEMBASEHI));
X       *mem <<= 8;
X
X       return 0;
}
X
static int msnd_activate_logical(int cfg, int num)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X       if (msnd_write_cfg(cfg, IREG_ACTIVATE, LD_ACTIVATE))
X               return -EIO;
X       return 0;
}
X
static int msnd_write_cfg_logical(int cfg, int num, WORD io0, WORD io1, WORD
irq, int mem)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X       if (msnd_write_cfg_io0(cfg, num, io0))
X               return -EIO;
X       if (msnd_write_cfg_io1(cfg, num, io1))
X               return -EIO;
X       if (msnd_write_cfg_irq(cfg, num, irq))
X               return -EIO;
X       if (msnd_write_cfg_mem(cfg, num, mem))
X               return -EIO;
X       if (msnd_activate_logical(cfg, num))
X               return -EIO;
X       return 0;
}
X
static int msnd_read_cfg_logical(int cfg, int num, WORD *io0, WORD *io1, WORD
*irq, int *mem)
{
X       if (msnd_write_cfg(cfg, IREG_LOGDEVICE, num))
X               return -EIO;
X       if (msnd_read_cfg_io0(cfg, num, io0))
X               return -EIO;
X       if (msnd_read_cfg_io1(cfg, num, io1))
X               return -EIO;
X       if (msnd_read_cfg_irq(cfg, num, irq))
X               return -EIO;
X       if (msnd_read_cfg_mem(cfg, num, mem))
X               return -EIO;
X       return 0;
}
```

```
X
static void usage(void)
{
X       fprintf(stderr,
X               "\n"
X               "pinnaclecfg 1.0\n"
X               "\n"
X               "usage: pinnaclecfg <config port> [device config]\n"
X               "\n"
X               "This is for use with the card in NON-PnP mode only.\n"
X               "\n"
X               "Available devices (not all available for Fiji):\n"
X               "\n"
X               "        Device                       Description\n"
X               "
--------------------------------------------------------------------\n"
X               "        reset                        Reset all devices (i.e.
disable)\n"
X               "        show                         Display current device
configurations\n"
X               "\n"
X               "        dsp <io> <irq> <mem>         Audio device\n"
X               "        mpu <io> <irq>               Internal Kurzweil synth\n"
X               "        ide <io0> <io1> <irq>        On-board IDE controller\n"
X               "        joystick <io>                Joystick port\n"
X               "\n");
X       exit(1);
}
X
static int cfg_reset(void)
{
X       int i;
X
X       for (i = 0; i < 4; ++i)
X               msnd_write_cfg_logical(config_port, i, 0, 0, 0, 0);
X
X       return 0;
}
X
static int cfg_show(void)
{
X       int i;
X       int count = 0;
X
X       for (i = 0; i < 4; ++i) {
X               WORD io0, io1, irq;
X               int mem;
X               msnd_read_cfg_logical(config_port, i, &io0, &io1, &irq, &mem);
X               switch (i) {
X               case 0:
X                       if (io0 || irq || mem) {
X                               printf("dsp 0x%x %d 0x%x\n", io0, irq, mem);
X                               ++count;
X                       }
X                       break;
X               case 1:
```

```
X                               if (io0 || irq) {
X                                       printf("mpu 0x%x %d\n", io0, irq);
X                                       ++count;
X                               }
X                               break;
X                       case 2:
X                               if (io0 || io1 || irq) {
X                                       printf("ide 0x%x 0x%x %d\n", io0, io1, irq);
X                                       ++count;
X                               }
X                               break;
X                       case 3:
X                               if (io0) {
X                                       printf("joystick 0x%x\n", io0);
X                                       ++count;
X                               }
X                               break;
X                       }
X               }
X
X               if (count == 0)
X                       fprintf(stderr, "no devices configured\n");
X
X               return 0;
}
X
static int cfg_dsp(int argc, char *argv[])
{
X               int io, irq, mem;
X
X               if (argc < 3 ||
X                   sscanf(argv[0], "0x%x", &io) != 1 ||
X                   sscanf(argv[1], "%d", &irq) != 1 ||
X                   sscanf(argv[2], "0x%x", &mem) != 1)
X                       usage();
X
X               if (!(io == 0x290 ||
X                     io == 0x260 ||
X                     io == 0x250 ||
X                     io == 0x240 ||
X                     io == 0x230 ||
X                     io == 0x220 ||
X                     io == 0x210 ||
X                     io == 0x3e0)) {
X                       fprintf(stderr, "error: io must be one of "
X                               "210, 220, 230, 240, 250, 260, 290, or 3E0\n");
X                       usage();
X               }
X
X               if (!(irq == 5 ||
X                     irq == 7 ||
X                     irq == 9 ||
X                     irq == 10 ||
X                     irq == 11 ||
X                     irq == 12)) {
X                       fprintf(stderr, "error: irq must be one of "
X                       "5, 7, 9, 10, 11 or 12\n");
X               usage();
X       }
X
X       if (!(mem == 0xb0000 ||
X             mem == 0xc8000 ||
X             mem == 0xd0000 ||
X             mem == 0xd8000 ||
X             mem == 0xe0000 ||
X             mem == 0xe8000)) {
X               fprintf(stderr, "error: mem must be one of "
X                       "0xb0000, 0xc8000, 0xd0000, 0xd8000, 0xe0000 or
0xe8000\n");
X               usage();
X       }
X
X       return msnd_write_cfg_logical(config_port, 0, io, 0, irq, mem);
}
X
static int cfg_mpu(int argc, char *argv[])
{
X       int io, irq;
X
X       if (argc < 2 ||
X           sscanf(argv[0], "0x%x", &io) != 1 ||
X           sscanf(argv[1], "%d", &irq) != 1)
X               usage();
X
X       return msnd_write_cfg_logical(config_port, 1, io, 0, irq, 0);
}
X
static int cfg_ide(int argc, char *argv[])
{
X       int io0, io1, irq;
X
X       if (argc < 3 ||
X           sscanf(argv[0], "0x%x", &io0) != 1 ||
X           sscanf(argv[0], "0x%x", &io1) != 1 ||
X           sscanf(argv[1], "%d", &irq) != 1)
X               usage();
X
X       return msnd_write_cfg_logical(config_port, 2, io0, io1, irq, 0);
}
X
static int cfg_joystick(int argc, char *argv[])
{
X       int io;
X
X       if (argc < 1 ||
X           sscanf(argv[0], "0x%x", &io) != 1)
X               usage();
X
X       return msnd_write_cfg_logical(config_port, 3, io, 0, 0, 0);
}
X
int main(int argc, char *argv[])
```

```
{
X       char *device;
X       int rv = 0;
X
X       --argc; ++argv;
X
X       if (argc < 2)
X               usage();
X
X       sscanf(argv[0], "0x%x", &config_port);
X       if (config_port != 0x250 && config_port != 0x260 && config_port !=
0x270) {
X               fprintf(stderr, "error: <config port> must be 0x250, 0x260 or
0x270\n");
X               exit(1);
X       }
X       if (ioperm(config_port, 2, 1)) {
X               perror("ioperm");
X               fprintf(stderr, "note: pinnaclecfg must be run as root\n");
X               exit(1);
X       }
X       device = argv[1];
X
X       argc -= 2; argv += 2;
X
X       if (strcmp(device, "reset") == 0)
X               rv = cfg_reset();
X       else if (strcmp(device, "show") == 0)
X               rv = cfg_show();
X       else if (strcmp(device, "dsp") == 0)
X               rv = cfg_dsp(argc, argv);
X       else if (strcmp(device, "mpu") == 0)
X               rv = cfg_mpu(argc, argv);
X       else if (strcmp(device, "ide") == 0)
X               rv = cfg_ide(argc, argv);
X       else if (strcmp(device, "joystick") == 0)
X               rv = cfg_joystick(argc, argv);
X       else {
X               fprintf(stderr, "error: unknown device %s\n", device);
X               usage();
X       }
X
X       if (rv)
X               fprintf(stderr, "error: device configuration failed\n");
X
X       return 0;
}
SHAR_EOF
  $shar_touch -am 1204092598 'MultiSound.d/pinnaclecfg.c' &&
  chmod 0664 'MultiSound.d/pinnaclecfg.c' ||
  $echo 'restore of' 'MultiSound.d/pinnaclecfg.c' 'failed'
  if ( md5sum --help 2>&1 | grep 'sage: md5sum \[' ) >/dev/null 2>&1 \
  && ( md5sum --version 2>&1 | grep -v 'textutils 1.12' ) >/dev/null; then
    md5sum -c << SHAR_EOF >/dev/null 2>&1 \
    || $echo 'MultiSound.d/pinnaclecfg.c:' 'MD5 check failed'
366bdf27f0db767a3c7921d0a6db20fe  MultiSound.d/pinnaclecfg.c
```

```
SHAR_EOF
  else
    shar_count="`LC_ALL= LC_CTYPE= LANG= wc -c < 'MultiSound.d/pinnaclecfg.c'`"
    test 10235 -eq "$shar_count" ||
    $echo 'MultiSound.d/pinnaclecfg.c:' 'original size' '10235,' 'current size'
"$shar_count!"
  fi
fi
# ============= MultiSound.d/Makefile ==============
if test -f 'MultiSound.d/Makefile' && test "$first_param" != -c; then
  $echo 'x -' SKIPPING 'MultiSound.d/Makefile' '(file already exists)'
else
  $echo 'x -' extracting 'MultiSound.d/Makefile' '(text)'
  sed 's/^X//' << 'SHAR_EOF' > 'MultiSound.d/Makefile' &&
CC      = gcc
CFLAGS  = -O
PROGS   = setdigital msndreset pinnaclecfg conv
X
all: $(PROGS)
X
clean:
X       rm -f $(PROGS)
SHAR_EOF
  $shar_touch -am 1204092398 'MultiSound.d/Makefile' &&
  chmod 0664 'MultiSound.d/Makefile' ||
  $echo 'restore of' 'MultiSound.d/Makefile' 'failed'
  if ( md5sum --help 2>&1 | grep 'sage: md5sum \[' ) >/dev/null 2>&1 \
  && ( md5sum --version 2>&1 | grep -v 'textutils 1.12' ) >/dev/null; then
    md5sum -c << SHAR_EOF >/dev/null 2>&1 \
    || $echo 'MultiSound.d/Makefile:' 'MD5 check failed'
76ca8bb44e3882edcf79c97df6c81845  MultiSound.d/Makefile
SHAR_EOF
  else
    shar_count="`LC_ALL= LC_CTYPE= LANG= wc -c < 'MultiSound.d/Makefile'`"
    test 106 -eq "$shar_count" ||
    $echo 'MultiSound.d/Makefile:' 'original size' '106,' 'current size'
"$shar_count!"
  fi
fi
# ============= MultiSound.d/conv.l ==============
if test -f 'MultiSound.d/conv.l' && test "$first_param" != -c; then
  $echo 'x -' SKIPPING 'MultiSound.d/conv.l' '(file already exists)'
else
  $echo 'x -' extracting 'MultiSound.d/conv.l' '(text)'
  sed 's/^X//' << 'SHAR_EOF' > 'MultiSound.d/conv.l' &&
%%
[ \n\t,\r]
\;.*
DB
[0-9A-Fa-f]+H   { int n; sscanf(yytext, "%xH", &n); printf("%c", n); }
%%
int yywrap() { return 1; }
main() { yylex(); }
SHAR_EOF
  $shar_touch -am 0828231798 'MultiSound.d/conv.l' &&
  chmod 0664 'MultiSound.d/conv.l' ||
  $echo 'restore of' 'MultiSound.d/conv.l' 'failed'
  if ( md5sum --help 2>&1 | grep 'sage: md5sum \[' ) >/dev/null 2>&1 \
  && ( md5sum --version 2>&1 | grep -v 'textutils 1.12' ) >/dev/null; then
    md5sum -c << SHAR_EOF >/dev/null 2>&1 \
    || $echo 'MultiSound.d/conv.l:' 'MD5 check failed'
d2411fc32cd71a00dcdc1f009e858dd2  MultiSound.d/conv.l
SHAR_EOF
  else
    shar_count="`LC_ALL= LC_CTYPE= LANG= wc -c < 'MultiSound.d/conv.l'`"
    test 141 -eq "$shar_count" ||
    $echo 'MultiSound.d/conv.l:' 'original size' '141,' 'current size'
"$shar_count!"
  fi
fi
# ============= MultiSound.d/msndreset.c ==============
if test -f 'MultiSound.d/msndreset.c' && test "$first_param" != -c; then
  $echo 'x -' SKIPPING 'MultiSound.d/msndreset.c' '(file already exists)'
else
  $echo 'x -' extracting 'MultiSound.d/msndreset.c' '(text)'
  sed 's/^X//' << 'SHAR_EOF' > 'MultiSound.d/msndreset.c' &&
/*********************************************************************
X *
X * msndreset.c - resets the MultiSound card
X *
X * Copyright (C) 1998 Andrew Veliath
X *
X * This program is free software; you can redistribute it and/or modify
X * it under the terms of the GNU General Public License as published by
X * the Free Software Foundation; either version 2 of the License, or
X * (at your option) any later version.
X *
X * This program is distributed in the hope that it will be useful,
X * but WITHOUT ANY WARRANTY; without even the implied warranty of
X * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
X * GNU General Public License for more details.
X *
X * You should have received a copy of the GNU General Public License
X * along with this program; if not, write to the Free Software
X * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
X *
X ********************************************************************/
X
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/soundcard.h>
X
int main(int argc, char *argv[])
{
X       int fd;
X
X       if (argc != 2) {
X               fprintf(stderr, "usage: msndreset <mixer device>\n");
X               exit(1);
X       }
X
X       if ((fd = open(argv[1], O_RDWR)) < 0) {
X               perror(argv[1]);
X               exit(1);
X       }
X
X       if (ioctl(fd, SOUND_MIXER_PRIVATE1, 0) < 0) {
X               fprintf(stderr, "error: msnd ioctl reset failed\n");
X               perror("ioctl");
X               close(fd);
X               exit(1);
X       }
X
X       close(fd);
X
X       return 0;
}
SHAR_EOF
  $shar_touch -am 1204100698 'MultiSound.d/msndreset.c' &&
  chmod 0664 'MultiSound.d/msndreset.c' ||
  $echo 'restore of' 'MultiSound.d/msndreset.c' 'failed'
  if ( md5sum --help 2>&1 | grep 'sage: md5sum \[' ) >/dev/null 2>&1 \
  && ( md5sum --version 2>&1 | grep -v 'textutils 1.12' ) >/dev/null; then
    md5sum -c << SHAR_EOF >/dev/null 2>&1 \
    || $echo 'MultiSound.d/msndreset.c:' 'MD5 check failed'
c52f876521084e8eb25e12e01dcccb8a  MultiSound.d/msndreset.c
SHAR_EOF
  else
    shar_count="`LC_ALL= LC_CTYPE= LANG= wc -c < 'MultiSound.d/msndreset.c'`"
    test 1472 -eq "$shar_count" ||
    $echo 'MultiSound.d/msndreset.c:' 'original size' '1472,' 'current size'
"$shar_count!"
  fi
fi
rm -fr _sh01426
exit 0
```