```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
   <TITLE>OSS Sequencer Emulation on ALSA</TITLE>
</HEAD>
<BODY>

<CENTER>
<H1>

<HR WIDTH="100%"></H1></CENTER>

<CENTER>
<H1>
OSS Sequencer Emulation on ALSA</H1></CENTER>

<HR WIDTH="100%">
<P>Copyright (c) 1998,1999 by Takashi Iwai
<TT><A
HREF="mailto:iwai@ww.uni-erlangen.de">&lt;iwai@ww.uni-erlangen.de></A></TT>
<P>ver.0.1.8; Nov. 16, 1999
<H2>

<HR WIDTH="100%"></H2>

<H2>
1. Description</H2>
This directory contains the OSS sequencer emulation driver on ALSA. Note
that this program is still in the development state.
<P>What this does - it provides the emulation of the OSS sequencer, access
via
<TT>/dev/sequencer</TT> and <TT>/dev/music</TT> devices.
The most of applications using OSS can run if the appropriate ALSA
sequencer is prepared.
<P>The following features are emulated by this driver:
<UL>
<LI>
Normal sequencer and MIDI events:</LI>

<BR>They are converted to the ALSA sequencer events, and sent to the
corresponding
port.
<LI>
Timer events:</LI>

<BR>The timer is not selectable by ioctl. The control rate is fixed to
100 regardless of HZ. That is, even on Alpha system, a tick is always
1/100 second. The base rate and tempo can be changed in <TT>/dev/music</TT>.

<LI>
Patch loading:</LI>

<BR>It purely depends on the synth drivers whether it's supported since
the patch loading is realized by callback to the synth driver.
<LI>
I/O controls:</LI>
```

```
<BR>Most of controls are accepted. Some controls
are dependent on the synth driver, as well as even on original OSS.</UL>
Furthermore, you can find the following advanced features:
<UL>
<LI>
Better queue mechanism:</LI>

<BR>The events are queued before processing them.
<LI>
Multiple applications:</LI>

<BR>You can run two or more applications simultaneously (even for OSS
sequencer)!
However, each MIDI device is exclusive - that is, if a MIDI device is opened
once by some application, other applications can’t use it. No such a restriction
in synth devices.
<LI>
Real-time event processing:</LI>

<BR>The events can be processed in real time without using out of bound
ioctl. To switch to real-time mode, send ABSTIME 0 event. The followed
events will be processed in real-time without queued. To switch off the
real-time mode, send RELTIME 0 event.
<LI>
<TT>/proc</TT> interface:</LI>

<BR>The status of applications and devices can be shown via
<TT>/proc/asound/seq/oss</TT>
at any time. In the later version, configuration will be changed via
<TT>/proc</TT>
interface, too.</UL>

<H2>
2. Installation</H2>
Run configure script with both sequencer support (<TT>--with-sequencer=yes</TT>)
and OSS emulation (<TT>--with-oss=yes</TT>) options. A module
<TT>snd-seq-oss.o</TT>
will be created. If the synth module of your sound card supports for OSS
emulation (so far, only Emu8000 driver), this module will be loaded
automatically.
Otherwise, you need to load this module manually.
<P>At beginning, this module probes all the MIDI ports which have been
already connected to the sequencer. Once after that, the creation and deletion
of ports are watched by announcement mechanism of ALSA sequencer.
<P>The available synth and MIDI devices can be found in proc interface.
Run "<TT>cat /proc/asound/seq/oss</TT>", and check the devices. For example,
if you use an AWE64 card, you’ll see like the following:
<PRE>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; OSS sequencer emulation version
0.1.8
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; ALSA client number 63
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; ALSA receiver port 0

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Number of applications: 0

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Number of synth devices: 1
```

```
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; synth 0: [EMU8000]
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; type 0x1 : subtype 0x20 : voices 32
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; capabilties : ioctl enabled / load_patch enabled

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Number of MIDI devices: 3

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; midi 0: [Emu8000 Port-0] ALSA port 65:0
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; capability write / opened none

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; midi 1: [Emu8000 Port-1] ALSA port 65:1
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; capability write / opened none

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; midi 2: [0: MPU-401 (UART)] ALSA port 64:0
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; capability read/write / opened none</PRE>
Note that the device number may be different from the information of
<TT>/proc/asound/oss-devices</TT>
or ones of the original OSS driver. Use the device number listed in
<TT>/proc/asound/seq/oss</TT>
to play via OSS sequencer emulation.
<H2>
3. Using Synthesizer Devices</H2>
Run your favorite program. I've tested playmidi-2.4, awemidi-0.4.3, gmod-3.1
and xmp-1.1.5. You can load samples via <TT>/dev/sequencer</TT> like sfxload,
too.
<P>If the lowlevel driver supports multiple access to synth devices (like
Emu8000 driver), two or more applications are allowed to run at the same
time.
<H2>
4. Using MIDI Devices</H2>
So far, only MIDI output was tested. MIDI input was not checked at all,
but hopefully it will work. Use the device number listed in
<TT>/proc/asound/seq/oss</TT>.
Be aware that these numbers are mostly different from the list in
<TT>/proc/asound/oss-devices</TT>.
<H2>
5. Module Options</H2>
The following module options are available:
<UL>
<LI>
<TT>maxqlen</TT></LI>

<BR>specifies the maximum read/write queue length. This queue is private
for OSS sequencer, so that it is independent from the queue length of ALSA
sequencer. Default value is 1024.
<LI>
<TT>seq_oss_debug</TT></LI>
```

```
<BR>specifies the debug level and accepts zero (= no debug message) or
positive integer. Default value is 0.</UL>

<H2>
6. Queue Mechanism</H2>
OSS sequencer emulation uses an ALSA priority queue. The
events from <TT>/dev/sequencer</TT> are processed and put onto the queue
specified by module option.
<P>All the events from <TT>/dev/sequencer</TT> are parsed at beginning.
The timing events are also parsed at this moment, so that the events may
be processed in real-time. Sending an event ABSTIME 0 switches the operation
mode to real-time mode, and sending an event RELTIME 0 switches it off.
In the real-time mode, all events are dispatched immediately.
<P>The queued events are dispatched to the corresponding ALSA sequencer
ports after scheduled time by ALSA sequencer dispatcher.
<P>If the write-queue is full, the application sleeps until a certain amount
(as default one half) becomes empty in blocking mode. The synchronization
to write timing was implemented, too.
<P>The input from MIDI devices or echo-back events are stored on read FIFO
queue. If application reads <TT>/dev/sequencer</TT> in blocking mode, the
process will be awaked.

<H2>
7. Interface to Synthesizer Device</H2>

<H3>
7.1. Registration</H3>
To register an OSS synthesizer device, use <TT>snd_seq_oss_synth_register</TT>
function.
<PRE>int snd_seq_oss_synth_register(char *name, int type, int subtype, int nvoices,
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; snd_seq_oss_callback_t *oper, void *private_data)</PRE>
The arguments <TT>name</TT>, <TT>type</TT>, <TT>subtype</TT> and
<TT>nvoices</TT>
are used for making the appropriate synth_info structure for ioctl. The
return value is an index number of this device. This index must be remembered
for unregister. If registration is failed, -errno will be returned.
<P>To release this device, call <TT>snd_seq_oss_synth_unregister function</TT>:
<PRE>int snd_seq_oss_synth_unregister(int index),</PRE>
where the <TT>index</TT> is the index number returned by register function.
<H3>
7.2. Callbacks</H3>
OSS synthesizer devices have capability for sample downloading and ioctls
like sample reset. In OSS emulation, these special features are realized
by using callbacks. The registration argument oper is used to specify these
callbacks. The following callback functions must be defined:
<PRE>snd_seq_oss_callback_t:
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int (*open)(snd_seq_oss_arg_t *p, void *closure);
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int (*close)(snd_seq_oss_arg_t *p);
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int (*ioctl)(snd_seq_oss_arg_t *p, unsigned int cmd, unsigned long arg);
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int (*load_patch)(snd_seq_oss_arg_t *p, int format, const char *buf, int offs, int count);
```

```
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int (*reset)(snd_seq_oss_arg_t *p);
Except for <TT>open</TT> and <TT>close</TT> callbacks, they are allowed
to be NULL.
<P>Each callback function takes the argument type snd_seq_oss_arg_t as the
first argument.
<PRE>struct snd_seq_oss_arg_t {
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int app_index;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int file_mode;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int seq_mode;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; snd_seq_addr_t addr;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; void *private_data;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; int event_passing;
};</PRE>
The first three fields, <TT>app_index</TT>, <TT>file_mode</TT> and
<TT>seq_mode</TT>
are initialized by OSS sequencer. The <TT>app_index</TT> is the application
index which is unique to each application opening OSS sequencer. The
<TT>file_mode</TT>
is bit-flags indicating the file operation mode. See
<TT>seq_oss.h</TT>
for its meaning. The <TT>seq_mode</TT> is sequencer operation mode. In
the current version, only <TT>SND_OSSSEQ_MODE_SYNTH</TT> is used.
<P>The next two fields, <TT>addr</TT> and <TT>private_data</TT>, must be
filled by the synth driver at open callback. The <TT>addr</TT> contains
the address of ALSA sequencer port which is assigned to this device. If
the driver allocates memory for <TT>private_data</TT>, it must be released
in close callback by itself.
<P>The last field, <TT>event_passing</TT>, indicates how to translate note-on
/ off events. In <TT>PROCESS_EVENTS</TT> mode, the note 255 is regarded
as velocity change, and key pressure event is passed to the port. In
<TT>PASS_EVENTS</TT>
mode, all note on/off events are passed to the port without modified.
<TT>PROCESS_KEYPRESS</TT>
mode checks the note above 128 and regards it as key pressure event (mainly
for Emu8000 driver).
<H4>
7.2.1. Open Callback</H4>
The <TT>open</TT> is called at each time this device is opened by an application
using OSS sequencer. This must not be NULL. Typically, the open callback
does the following procedure:
<OL>
<LI>
Allocate private data record.</LI>

<LI>
Create an ALSA sequencer port.</LI>

<LI>
Set the new port address on arg->addr.</LI>

<LI>
Set the private data record pointer on arg->private_data.</LI>
</OL>
Note that the type bit-flags in port_info of this synth port must NOT contain
<TT>TYPE_MIDI_GENERIC</TT>
bit. Instead, <TT>TYPE_SPECIFIC</TT> should be used. Also,
<TT>CAP_SUBSCRIPTION</TT>
bit should NOT be included, too. This is necessary to tell it from other
normal MIDI devices. If the open procedure succeeded, return zero. Otherwise,
return -errno.
<H4>
7.2.2 Ioctl Callback</H4>
The <TT>ioctl</TT> callback is called when the sequencer receives
device-specific
ioctls. The following two ioctls should be processed by this callback:
<UL>
<LI>
<TT>IOCTL_SEQ_RESET_SAMPLES</TT></LI>

<BR>reset all samples on memory -- return 0
<LI>
<TT>IOCTL_SYNTH_MEMAVL</TT></LI>

<BR>return the available memory size
<LI>
<TT>FM_4OP_ENABLE</TT></LI>

<BR>can be ignored usually</UL>
The other ioctls are processed inside the sequencer without passing to
the lowlevel driver.
<H4>
7.2.3 Load_Patch Callback</H4>
The <TT>load_patch</TT> callback is used for sample-downloading. This callback
must read the data on user-space and transfer to each device. Return 0
if succeeded, and -errno if failed. The format argument is the patch key
in patch_info record. The buf is user-space pointer where patch_info record
is stored. The offs can be ignored. The count is total data size of this
sample data.
<H4>
7.2.4 Close Callback</H4>
The <TT>close</TT> callback is called when this device is closed by the
applicaion. If any private data was allocated in open callback, it must
be released in the close callback. The deletion of ALSA port should be
done here, too. This callback must not be NULL.
<H4>
7.2.5 Reset Callback</H4>
The <TT>reset</TT> callback is called when sequencer device is reset or
closed by applications. The callback should turn off the sounds on the
relevant port immediately, and initialize the status of the port. If this
callback is undefined, OSS seq sends a <TT>HEARTBEAT</TT> event to the
port.
<H3>
7.3 Events</H3>
Most of the events are processed by sequencer and translated to the adequate
ALSA sequencer events, so that each synth device can receive by input_event
callback of ALSA sequencer port. The following ALSA events should be implemented
by the driver:
<BR>&nbsp;
<TABLE BORDER WIDTH="75%" NOSAVE >
<TR NOSAVE>
<TD NOSAVE><B>ALSA event</B></TD>
```

```
<TD><B>Original OSS events</B></TD>
</TR>

<TR>
<TD>NOTEON</TD>

<TD>SEQ_NOTEON
<BR>MIDI_NOTEON</TD>
</TR>

<TR>
<TD>NOTE</TD>

<TD>SEQ_NOTEOFF
<BR>MIDI_NOTEOFF</TD>
</TR>

<TR NOSAVE>
<TD NOSAVE>KEYPRESS</TD>

<TD>MIDI_KEY_PRESSURE</TD>
</TR>

<TR NOSAVE>
<TD>CHANPRESS</TD>

<TD NOSAVE>SEQ_AFTERTOUCH
<BR>MIDI_CHN_PRESSURE</TD>
</TR>

<TR NOSAVE>
<TD NOSAVE>PGMCHANGE</TD>

<TD NOSAVE>SEQ_PGMCHANGE
<BR>MIDI_PGM_CHANGE</TD>
</TR>

<TR>
<TD>PITCHBEND</TD>

<TD>SEQ_CONTROLLER(CTRL_PITCH_BENDER)
<BR>MIDI_PITCH_BEND</TD>
</TR>

<TR>
<TD>CONTROLLER</TD>

<TD>MIDI_CTL_CHANGE
<BR>SEQ_BALANCE (with CTL_PAN)</TD>
</TR>

<TR>
<TD>CONTROL14</TD>

<TD>SEQ_CONTROLLER</TD>
</TR>
```

```
<TR>
<TD>REGPARAM</TD>

<TD>SEQ_CONTROLLER(CTRL_PITCH_BENDER_RANGE)</TD>
</TR>

<TR>
<TD>SYSEX</TD>

<TD>SEQ_SYSEX</TD>
</TR>
</TABLE>

<P>The most of these behavior can be realized by MIDI emulation driver
included in the Emu8000 lowlevel driver. In the future release, this module
will be independent.
<P>Some OSS events (<TT>SEQ_PRIVATE</TT> and <TT>SEQ_VOLUME</TT> events) are
passed as event
type SND_SEQ_OSS_PRIVATE.  The OSS sequencer passes these event 8 byte
packets without any modification. The lowlevel driver should process these
events appropriately.
<H2>
8. Interface to MIDI Device</H2>
Since the OSS emulation probes the creation and deletion of ALSA MIDI sequencer
ports automatically by receiving announcement from ALSA sequencer, the
MIDI devices don't need to be registered explicitly like synth devices.
However, the MIDI port_info registered to ALSA sequencer must include a group
name <TT>SND_SEQ_GROUP_DEVICE</TT> and a capability-bit <TT>CAP_READ</TT> or
<TT>CAP_WRITE</TT>. Also, subscription capabilities, <TT>CAP_SUBS_READ</TT> or
<TT>CAP_SUBS_WRITE</TT>,
must be defined, too. If these conditions are not satisfied, the port is not
registered as OSS sequencer MIDI device.
<P>The events via MIDI devices are parsed in OSS sequencer and converted
to the corresponding ALSA sequencer events. The input from MIDI sequencer
is also converted to MIDI byte events by OSS sequencer. This works just
a reverse way of seq_midi module.
<H2>
9. Known Problems / TODO's</H2>

<UL>
<LI>
Patch loading via ALSA instrument layer is not implemented yet.</LI>
</UL>

</BODY>
</HTML>
```