ALSA PCM Timestamping

The ALSA API can provide two different system timestamps:

- Trigger_tstamp is the system time snapshot taken when the .trigger callback is invoked. This snapshot is taken by the ALSA core in the general case, but specific hardware may have synchronization capabilities or conversely may only be able to provide a correct estimate with a delay. In the latter two cases, the low-level driver is responsible for updating the trigger_tstamp at the most appropriate and precise moment. Applications should not rely solely on the first trigger_tstamp but update their internal calculations if the driver provides a refined estimate with a delay.
- tstamp is the current system timestamp updated during the last event or application query. The difference (tstamp - trigger_tstamp) defines the elapsed time.

The ALSA API provides two basic pieces of information, avail and delay, which combined with the trigger and current system timestamps allow for applications to keep track of the 'fullness' of the ring buffer and the amount of queued samples.

The use of these different pointers and time information depends on the application needs:

- `avail` reports how much can be written in the ring buffer
- `delay` reports the time it will take to hear a new sample after all queued samples have been played out.

When timestamps are enabled, the avail/delay information is reported along with a snapshot of system time. Applications can select from `CLOCK_REALTIME` (NTP corrections including going backwards), `CLOCK_MONOTONIC` (NTP corrections but never going backwards), `CLOCK_MONOTIC_RAW` (without NTP corrections) and change the mode dynamically with sw_params

The ALSA API also provide an audio_tstamp which reflects the passage of time as measured by different components of audio hardware. In ascii-art, this could be represented as follows (for the playback case):

```
--------------------------------------------------------------> time
  ^               ^              ^                ^           ^
  |               |              |                |           |
 analog         link            dma              app       FullBuffer
 time           time           time              time        time
  |               |              |                |           |
  |< codec delay >|<--hw delay-->|<queued samples>|<---avail->|
  |<----------------- delay---------------------->|           |
                                 |<----ring buffer length---->|
```

The analog time is taken at the last stage of the playback, as close as possible to the actual transducer

The link time is taken at the output of the SoC/chipset as the samples are pushed on a link. The link time can be directly measured if supported in hardware by sample counters or wallclocks (e.g. with HDAudio 24MHz or PTP clock for networked solutions) or indirectly estimated (e.g. with the frame counter in USB).

The DMA time is measured using counters - typically the least reliable of all measurements due to the bursty nature of DMA transfers.

The app time corresponds to the time tracked by an application after writing in the ring buffer.

The application can query the hardware capabilities, define which audio time it wants reported by selecting the relevant settings in audio_tstamp_config fields, thus get an estimate of the timestamp accuracy. It can also request the delay-to-analog be included in the measurement. Direct access to the link time is very interesting on platforms that provide an embedded DSP; measuring directly the link time with dedicated hardware, possibly synchronized with system time, removes the need to keep track of internal DSP processing times and latency.

In case the application requests an audio tstamp that is not supported in hardware/low-level driver, the type is overridden as DEFAULT and the timestamp will report the DMA time based on the hw_pointer value.

For backwards compatibility with previous implementations that did not provide timestamp selection, with a zero-valued COMPAT timestamp type the results will default to the HDAudio wall clock for playback streams and to the DMA time (hw_ptr) in all other cases.

The audio timestamp accuracy can be returned to user-space, so that appropriate decisions are made:

- for dma time (default), the granularity of the transfers can be inferred from the steps between updates and in turn provide information on how much the application pointer can be rewound safely.
- the link time can be used to track long-term drifts between audio and system time using the (tstamp-trigger_tstamp)/audio_tstamp ratio, the precision helps define how much smoothing/low-pass filtering is required. The link time can be either reset on startup or reported as is (the latter being useful to compare progress of different streams - but may require the wallclock to be always running and not wrap-around during idle periods). If supported in hardware, the absolute link time could also be used to define a precise start time (patches WIP)
- including the delay in the audio timestamp may counter-intuitively not increase the precision of timestamps, e.g. if a codec includes variable-latency DSP processing or a chain of hardware components the delay is typically not known with precision.

The accuracy is reported in nanosecond units (using an unsigned 32-bit word), which gives a max precision of 4.29s, more than enough for audio applications...

Due to the varied nature of timestamping needs, even for a single application, the audio_tstamp_config can be changed dynamically. In the `STATUS` ioctl, the parameters are read-only and do not allow for any application selection. To work around this limitation without impacting legacy applications, a new `STATUS_EXT` ioctl is introduced with read/write parameters. ALSA-lib will be modified to make use of `STATUS_EXT` and effectively deprecate `STATUS`.

The ALSA API only allows for a single audio timestamp to be reported at a time. This is a conscious design decision, reading the audio timestamps from hardware registers or from IPC takes time, the more timestamps are read the more imprecise the combined measurements are. To avoid any interpretation issues, a single (system, audio) timestamp is reported. Applications that need different timestamps will be required to issue multiple queries and perform an interpolation of the results

In some hardware-specific configuration, the system timestamp is latched by a low-level audio subsystem, and the information provided back to the driver. Due to potential delays in the communication with the hardware, there is a risk of misalignment with the avail and delay information. To make sure applications are not confused, a driver_timestamp field is added in the snd_pcm_status structure; this timestamp shows when the information is put together by the driver before returning from the `STATUS` and `STATUS_EXT` ioctl. in most cases this driver_timestamp will be identical to the regular system tstamp.

Examples of timestamping with HDAudio:

1. DMA timestamp, no compensation for DMA+analog delay

```
$ ./audio_time  -p --ts_type=1
playback: systime: 341121338 nsec, audio time 342000000 nsec,         systime delta -878662
playback: systime: 426236663 nsec, audio time 427187500 nsec,         systime delta -950837
playback: systime: 597080580 nsec, audio time 598000000 nsec,         systime delta -919420
playback: systime: 682059782 nsec, audio time 683020833 nsec,         systime delta -961051
playback: systime: 852896415 nsec, audio time 853854166 nsec,         systime delta -957751
playback: systime: 937903344 nsec, audio time 938854166 nsec,         systime delta -950822
```

2. DMA timestamp, compensation for DMA+analog delay

```
$ ./audio_time  -p --ts_type=1 -d
playback: systime: 341053347 nsec, audio time 341062500 nsec,         systime delta -9153
playback: systime: 426072447 nsec, audio time 426062500 nsec,         systime delta 9947
playback: systime: 596899518 nsec, audio time 596895833 nsec,         systime delta 3685
playback: systime: 681915317 nsec, audio time 681916666 nsec,         systime delta -1349
playback: systime: 852741306 nsec, audio time 852750000 nsec,         systime delta -8694
```

3. link timestamp, compensation for DMA+analog delay

```
$ ./audio_time  -p --ts_type=2 -d
playback: systime: 341060004 nsec, audio time 341062791 nsec,         systime delta -2787
playback: systime: 426242074 nsec, audio time 426244875 nsec,         systime delta -2801
playback: systime: 597080992 nsec, audio time 597084583 nsec,         systime delta -3591
playback: systime: 682084512 nsec, audio time 682088291 nsec,         systime delta -3779
playback: systime: 852936229 nsec, audio time 852940916 nsec,         systime delta -4687
playback: systime: 938107562 nsec, audio time 938112708 nsec,         systime delta -5146
```

Example 1 shows that the timestamp at the DMA level is close to 1ms ahead of the actual playback time (as a side time this sort of measurement can help define rewind safeguards). Compensating for the DMA-link delay in example 2 helps remove the hardware buffering but the information is still very jittery, with up to one sample of error. In example 3 where the timestamps are measured with the link wallclock, the timestamps show a monotonic behavior and a lower dispersion.

Example 3 and 4 are with USB audio class. Example 3 shows a high offset between audio time and system time due to buffering. Example 4 shows how compensating for the delay exposes a 1ms accuracy (due to the use of the frame counter by the driver)

Example 3: DMA timestamp, no compensation for delay, delta of ~5ms

```
$ ./audio_time -p -Dhw:1 -t1
playback: systime: 120174019 nsec, audio time 125000000 nsec,         systime delta -4825981
playback: systime: 245041136 nsec, audio time 250000000 nsec,         systime delta -4958864
playback: systime: 370106088 nsec, audio time 375000000 nsec,         systime delta -4893912
playback: systime: 495040065 nsec, audio time 500000000 nsec,         systime delta -4959935
playback: systime: 620038179 nsec, audio time 625000000 nsec,         systime delta -4961821
playback: systime: 745087741 nsec, audio time 750000000 nsec,         systime delta -4912259
playback: systime: 870037336 nsec, audio time 875000000 nsec,         systime delta -4962664
```

Example 4: DMA timestamp, compensation for delay, delay of ~1ms

```
$ ./audio_time -p -Dhw:1 -t1 -d
playback: systime: 120190520 nsec, audio time 120000000 nsec,         systime delta 190520
playback: systime: 245036740 nsec, audio time 244000000 nsec,         systime delta 1036740
playback: systime: 370034081 nsec, audio time 369000000 nsec,         systime delta 1034081
playback: systime: 495159907 nsec, audio time 494000000 nsec,         systime delta 1159907
playback: systime: 620098824 nsec, audio time 619000000 nsec,         systime delta 1098824
playback: systime: 745031847 nsec, audio time 744000000 nsec,         systime delta 1031847
```