Tracepoints in ALSA

2017/07/02 Takasahi Sakamoto

Tracepoints in ALSA PCM core

ALSA PCM core registers `snd_pcm` subsystem to kernel tracepoint system. This subsystem includes two categories of tracepoints; for state of PCM buffer and for processing of PCM hardware parameters. These tracepoints are available when corresponding kernel configurations are enabled. When `CONFIG_SND_DEBUG` is enabled, the latter tracepoints are available. When additional `SND_PCM_XRUN_DEBUG` is enabled too, the former trace points are enabled.

Tracepoints for state of PCM buffer

This category includes four tracepoints; `hwptr`, `applptr`, `xrun` and `hw_ptr_error`.

Tracepoints for processing of PCM hardware parameters

This category includes two tracepoints; `hw_mask_param` and `hw_interval_param`.

In a design of ALSA PCM core, data transmission is abstracted as PCM substream. Applications manage PCM substream to maintain data transmission for PCM frames. Before starting the data transmission, applications need to configure PCM substream. In this procedure, PCM hardware parameters are decided by interaction between applications and ALSA PCM core. Once decided, runtime of the PCM substream keeps the parameters.

The parameters are described in struct snd_pcm_hw_params. This structure includes several types of parameters. Applications set preferable value to these parameters, then execute ioctl(2) with SNDRV_PCM_IOCTL_HW_REFINE or SNDRV_PCM_IOCTL_HW_PARAMS. The former is used just for refining available set of parameters. The latter is used for an actual decision of the parameters.

The struct snd_pcm_hw_params structure has below members:

`flags`
: Configurable. ALSA PCM core and some drivers handle this flag to select convenient parameters or change their behaviour.

`masks`
: Configurable. This type of parameter is described in struct snd_mask and represent mask values. As of PCM protocol v2.0.13, three types are defined.
  - SNDRV_PCM_HW_PARAM_ACCESS
  - SNDRV_PCM_HW_PARAM_FORMAT
  - SNDRV_PCM_HW_PARAM_SUBFORMAT

`intervals`
: Configurable. This type of parameter is described in struct snd_interval and represent values with a range. As of PCM protocol v2.0.13, twelve types are defined.
  - SNDRV_PCM_HW_PARAM_SAMPLE_BITS
  - SNDRV_PCM_HW_PARAM_FRAME_BITS
  - SNDRV_PCM_HW_PARAM_CHANNELS
  - SNDRV_PCM_HW_PARAM_RATE
  - SNDRV_PCM_HW_PARAM_PERIOD_TIME
  - SNDRV_PCM_HW_PARAM_PERIOD_SIZE
  - SNDRV_PCM_HW_PARAM_PERIOD_BYTES
  - SNDRV_PCM_HW_PARAM_PERIODS
  - SNDRV_PCM_HW_PARAM_BUFFER_TIME
  - SNDRV_PCM_HW_PARAM_BUFFER_SIZE
  - SNDRV_PCM_HW_PARAM_BUFFER_BYTES
  - SNDRV_PCM_HW_PARAM_TICK_TIME

`rmask`
: Configurable. This is evaluated at ioctl(2) with SNDRV_PCM_IOCTL_HW_REFINE only. Applications can select which mask/interval parameter can be changed by ALSA PCM core. For SNDRV_PCM_IOCTL_HW_PARAMS, this mask is ignored and all of parameters are going to be changed.

`cmask`
: Read-only. After returning from ioctl(2), buffer in user space for struct snd_pcm_hw_params includes result of each operation. This mask represents which mask/interval parameter is actually changed.

`info`
: Read-only. This represents hardware/driver capabilities as bit flags with SNDRV_PCM_INFO_XXX. Typically, applications execute ioctl(2) with SNDRV_PCM_IOCTL_HW_REFINE to retrieve this flag, then decide candidates of parameters and execute ioctl(2) with SNDRV_PCM_IOCTL_HW_PARAMS to configure PCM substream.

`msbits`
: Read-only. This value represents available bit width in MSB side of a PCM sample. When a parameter of SNDRV_PCM_HW_PARAM_SAMPLE_BITS was decided as a fixed number, this value is also calculated according to it. Else, zero. But this behaviour depends on implementations in driver side.

`rate_num`
: Read-only. This value represents numerator of sampling rate in fraction notation. Basically, when a parameter of SNDRV_PCM_HW_PARAM_RATE was decided as a single value, this value is also calculated according to it. Else, zero. But this behaviour depends on implementations in driver side.

`rate_den`
: Read-only. This value represents denominator of sampling rate in fraction notation. Basically, when a parameter of SNDRV_PCM_HW_PARAM_RATE was decided as a single value, this value is also calculated according to it. Else, zero. But this behaviour depends on implementations in driver side.

`fifo_size`
: Read-only. This value represents the size of FIFO in serial sound interface of hardware. Basically, each driver can assigns a proper value to this parameter but some drivers intentionally set zero with a care of hardware design or data transmission protocol.

ALSA PCM core handles buffer of struct snd_pcm_hw_params when applications execute ioctl(2) with SNDRV_PCM_HW_REFINE or SNDRV_PCM_HW_PARAMS. Parameters in the buffer are changed according to struct snd_pcm_hardware and rules of constraints in the runtime. The structure describes capabilities of handled hardware. The rules describes dependencies on which a parameter is decided according to several parameters. A rule has a callback function, and drivers can register arbitrary functions to compute the target parameter. ALSA PCM core registers some rules to the runtime as a default.

Each driver can join in the interaction as long as it prepared for two stuffs in a callback of struct snd_pcm_ops.open.

1. In the callback, drivers are expected to change a member of struct snd_pcm_hardware type in the runtime, according to capacities of corresponding hardware.
2. In the same callback, drivers are also expected to register additional rules of constraints into the runtime when several parameters have dependencies due to hardware design.

The driver can refers to result of the interaction in a callback of struct snd_pcm_ops.hw_params, however it should not change the content.

Tracepoints in this category are designed to trace changes of the mask/interval parameters. When ALSA PCM core changes them, `hw_mask_param` or `hw_interval_param` event is probed according to type of the changed parameter.

ALSA PCM core also has a pretty print format for each of the tracepoints. Below is an example for `hw_mask_param`.

```
hw_mask_param: pcmC0D0p 001/023 FORMAT 00000000000000000000001000000044 00000000000000000000001000000044
```

Below is an example for `hw_interval_param`.

```
hw_interval_param: pcmC0D0p 000/023 BUFFER_SIZE 0 0 [0 4294967295] 0 1 [0 4294967295]
```

The first three fields are common. They represent name of ALSA PCM character device, rules of constraint and name of the changed parameter, in order. The field for rules of constraint consists of two sub-fields; index of applied rule and total number of rules added to the runtime. As an exception, the index 000 means that the parameter is changed by ALSA PCM core, regardless of the rules.

The rest of field represent state of the parameter before/after changing. These fields are different according to type of the parameter. For parameters of mask type, the fields represent hexadecimal dump of content of the parameter. For parameters of interval type, the fields represent values of each member of `empty`, `integer`, `openmin`, `min`, `max`, `openmax` in struct snd_interval in this order.

Tracepoints in drivers

Some drivers have tracepoints for developers' convenience. For them, please refer to each documentation or implementation.