ASoC jack detection

ALSA has a standard API for representing physical jacks to user space, the kernel side of which can be seen in include/sound/jack.h. ASoC provides a version of this API adding two additional features:

- It allows more than one jack detection method to work together on one user visible jack. In embedded systems it is common for multiple to be present on a single jack but handled by separate bits of hardware.
- Integration with DAPM, allowing DAPM endpoints to be updated automatically based on the detected jack status (eg, turning off the headphone outputs if no headphones are present).

This is done by splitting the jacks up into three things working together: the jack itself represented by a struct snd_soc_jack, sets of snd_soc_jack_pins representing DAPM endpoints to update and blocks of code providing jack reporting mechanisms.

For example, a system may have a stereo headset jack with two reporting mechanisms, one for the headphone and one for the microphone. Some systems won't be able to use their speaker output while a headphone is connected and so will want to make sure to update both speaker and headphone when the headphone jack status changes.

The jack - struct snd_soc_jack

This represents a physical jack on the system and is what is visible to user space. The jack itself is completely passive, it is set up by the machine driver and updated by jack detection methods.

Jacks are created by the machine driver calling snd_soc_jack_new().

snd_soc_jack_pin

These represent a DAPM pin to update depending on some of the status bits supported by the jack. Each snd_soc_jack has zero or more of these which are updated automatically. They are created by the machine driver and associated with the jack using snd_soc_jack_add_pins(). The status of the endpoint may configured to be the opposite of the jack status if required (eg, enabling a built in microphone if a microphone is not connected via a jack).

Jack detection methods

Actual jack detection is done by code which is able to monitor some input to the system and update a jack by calling snd_soc_jack_report(), specifying a subset of bits to update. The jack detection code should be set up by the machine driver, taking configuration for the jack to update and the set of things to report when the jack is connected.

Often this is done based on the status of a GPIO - a handler for this is provided by the snd_soc_jack_add_gpio() function. Other methods are also available, for example integrated into CODECs. One example of CODEC integrated jack detection can be see in the WM8350 driver.

Each jack may have multiple reporting mechanisms, though it will need at least one to be useful.

Machine drivers

These are all hooked together by the machine driver depending on the system hardware. The machine driver will set up the snd_soc_jack and the list of pins to update then set up one or more jack detection mechanisms to update that jack based on their current status.