Realtek PC Beep Hidden Register

This file documents the "PC Beep Hidden Register", which is present in certain Realtek HDA codecs and controls a muxer and pair of passthrough mixers that can route audio between pins but aren't themselves exposed as HDA widgets. As far as I can tell, these hidden routes are designed to allow flexible PC Beep output for codecs that don't have mixer widgets in their output paths. Why it's easier to hide a mixer behind an undocumented vendor register than to just expose it as a widget, I have no idea.

Register Description

The register is accessed via processing coefficient 0x36 on NID 20h. Bits not identified below have no discernible effect on my machine, a Dell XPS 13 9350:

```
MSB                           LSB
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| |h|S|L|         | B |R|       | Known bits
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|0|0|1|1|  0x7  |0|0x0|1|  0x7  | Reset value
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

1Ah input select (B): 2 bits
: When zero, expose the PC Beep line (from the internal beep generator, when enabled with the Set Beep Generation verb on NID 01h, or else from the external PCBEEP pin) on the 1Ah pin node. When nonzero, expose the headphone jack (or possibly Line In on some machines) input instead. If PC Beep is selected, the 1Ah boost control has no effect.

Amplify 1Ah loopback, left (L): 1 bit
: Amplify the left channel of 1Ah before mixing it into outputs as specified by h and S bits. Does not affect the level of 1Ah exposed to other widgets.

Amplify 1Ah loopback, right (R): 1 bit
: Amplify the right channel of 1Ah before mixing it into outputs as specified by h and S bits. Does not affect the level of 1Ah exposed to other widgets.

Loopback 1Ah to 21h [active low] (h): 1 bit
: When zero, mix 1Ah (possibly with amplification, depending on L and R bits) into 21h (headphone jack on my machine). Mixed signal respects the mute setting on 21h.

Loopback 1Ah to 14h (S): 1 bit
: When one, mix 1Ah (possibly with amplification, depending on L and R bits) into 14h (internal speaker on my machine). Mixed signal **ignores** the mute setting on 14h and is present whenever 14h is configured as an output.

Path diagrams

1Ah input selection (DIV is the PC Beep divider set on NID 01h):

```
<Beep generator>   <PCBEEP pin>    <Headphone jack>
        |                |                |
        +--DIV--+--!DIV--+       {1Ah boost control}
                |                         |
                +--(b == 0)--+--(b != 0)--+
                             |
                >1Ah (Beep/Headphone Mic/Line In)<
```

Loopback of 1Ah to 21h/14h:

```
       <1Ah (Beep/Headphone Mic/Line In)>
                        |
                {amplify if L/R}
                        |
        +-----!h-----+-----S-----+
        |                        |
{21h mute control}               |
        |                        |
>21h (Headphone)<     >14h (Internal Speaker)<
```

Background

All Realtek HDA codecs have a vendor-defined widget with node ID 20h which provides access to a bank of registers that control various codec functions. Registers are read and written via the standard HDA processing coefficient verbs (Set/Get Coefficient Index, Set/Get Processing Coefficient). The node is named "Realtek Vendor Registers" in public datasheets' verb listings and, apart from that, is entirely undocumented.

This particular register, exposed at coefficient 0x36 and named in commits from Realtek, is of note: unlike most registers, which seem to control detailed amplifier parameters not in scope of the HDA specification, it controls audio routing which could just as easily

have been defined using standard HDA mixer and selector widgets.

Specifically, it selects between two sources for the input pin widget with Node ID (NID) 1Ah: the widget's signal can come either from an audio jack (on my laptop, a Dell XPS 13 9350, it's the headphone jack, but comments in Realtek commits indicate that it might be a Line In on some machines) or from the PC Beep line (which is itself multiplexed between the codec's internal beep generator and external PCBEEP pin, depending on if the beep generator is enabled via verbs on NID 01h). Additionally, it can mix (with optional amplification) that signal onto the 21h and/or 14h output pins.

The register's reset value is 0x3717, corresponding to PC Beep on 1Ah that is then amplified and mixed into both the headphones and the speakers. Not only does this violate the HDA specification, which says that "[a vendor defined beep input pin] connection may be maintained *only* while the Link reset (**RST#**) is asserted", it means that we cannot ignore the register if we care about the input that 1Ah would otherwise expose or if the PCBEEP trace is poorly shielded and picks up chassis noise (both of which are the case on my machine).

Unfortunately, there are lots of ways to get this register configuration wrong. Linux, it seems, has gone through most of them. For one, the register resets after S3 suspend: judging by existing code, this isn't the case for all vendor registers, and it's led to some fixes that improve behavior on cold boot but don't last after suspend. Other fixes have successfully switched the 1Ah input away from PC Beep but have failed to disable both loopback paths. On my machine, this means that the headphone input is amplified and looped back to the headphone output, which uses the exact same pins! As you might expect, this causes terrible headphone noise, the character of which is controlled by the 1Ah boost control. (If you've seen instructions online to fix XPS 13 headphone noise by changing "Headphone Mic Boost" in ALSA, now you know why.)

The information here has been obtained through black-box reverse engineering of the ALC256 codec's behavior and is not guaranteed to be correct. It likely also applies for the ALC255, ALC257, ALC235, and ALC236, since those codecs seem to be close relatives of the ALC256. (They all share one initialization function.) Additionally, other codecs like the ALC225 and ALC285 also have this register, judging by existing fixups in `patch_realtek.c`, but specific data (e.g. node IDs, bit positions, pin mappings) for those codecs may differ from what I've described here.