Contribution ID: **104** Type: **not specified**

”Exclusive” resources that are not exclusive - the case of shared GPIOs in DT systems

Saturday 13 December 2025 15:45 (20 minutes)

The high-level idea behind the linux kernel GPIO consumer API is that lines are an exclusive resource - only one logical consumer can request and control a GPIO pin. This results naturally from the type of operations that a low-level user can perform on GPIO lines - after all: one user setting the line’s direction to output while another sets it to input is an example of a very clear conflict that cannot easily be solved without involving a higher level abstraction wrapping the low-level GPIO primitives.

In real life however, nothing stops hardware designers from connecting the same IO pin to multiple devices and this is what we - as kernel engineers - later see modeled in device-tree as phandles referencing the same GPIO assigned to different device nodes.

In some cases, we have infrastructure in place to handle this correctly. An example would be: the semi-standardized reset-gpios DT property that will be interpreted by the reset core as a simple reset provider with a shared GPIO which is driven “high” when the reset is asserted. Reset core handles multiple references to the same “logical” reset.

Device-tree however is supposed to model the real hardware so as soon as the same GPIO is used by multiple devices in a different context, we are in a situation where two separate drivers need to request the same line. We currently only have a hack in place that allows drivers to basically “fight” over the GPIO with no concurrency protection and no actual logic behind.

That’s not all: the problem can become even more complex than a simple reference counting of value setting. An example of such a problem is the PERST# signal in PCI shared between endpoints where the expected (logical) behavior is: physically drive the line high only once ALL the users want it driven high.

This talk will describe the problems in detail, describe existing solutions and the need for better ones, what has been done so far and how much we can do in C code with the information contained in device-tree before we have to resort to quirks.

Primary author: Mr GOLASZEWSKI, Bartosz (Qualcomm)

Presenter: Mr GOLASZEWSKI, Bartosz (Qualcomm)

Session Classification: Devicetree MC

Track Classification: Devicetree MC