NAME

gpiobus - GPIO bus system

SYNOPSIS

To compile these devices into your kernel and use the device hints, place the following lines in your kernel configuration file:

```
device gpio
device gpioc
device gpioiic
device gpioled
```

Additional device entries for the ARM architecture include:

```
device a10_gpio
device bcm_gpio
device imx51_gpio
device lpcgpio
device mv_gpio
device ti_gpio
device gpio_avila
device gpio_cambria
device zy7_gpio
device pxagpio
```

Additional device entries for the MIPS architecture include:

```
device ar71xxx_gpio
device octeon_gpio
device rt305_gpio
```

Additional device entries for the POWERPC architecture include:

```
device wiigpio
device macgpio
```

Additional device entries for the RISC-V architecture include:

```
device sifive_gpio
```

DESCRIPTION

The **gpiobus** system provides a simple interface to the GPIO pins that are usually available on embedded architectures and can provide bit banging style devices to the system.

The acronym GPIO means "General-Purpose Input/Output."

The BUS physically consists of multiple pins that can be configured for input/output, IRQ delivery, SDA/SCL *iicbus* use, etc.

On some embedded architectures (like MIPS), discovery of the bus and configuration of the pins is done via device.hints(5) in the platform's kernel config(5) file.

On some others (like ARM), where FDT(4) is used to describe the device tree, the bus discovery is done via the DTS passed to the kernel, being either statically compiled in, or by a variety of ways where the boot loader (or Open Firmware enabled system) passes the DTS blob to the kernel at boot.

On a device.hints(5) based system these hints can be used to configure drivers for devices attached to **gpiobus** pins:

hint.driver.unit.at The **gpiobus** where the device is attached. For example, "gpiobus0". *driver* and *unit* are the driver name and the unit number for the device driver.

hint.driver.unit.pins This is a bitmask of the pins on the **gpiobus** that are connected to the device. The pins will be allocated to the specified driver instance. Only pins with numbers from 0 to 31 can be specified using this hint.

hint.driver.unit.pin_list This is a list of pin numbers of pins on the **gpiobus** that are connected to the device. The pins will be allocated to the specified driver instance. This is a more user friendly alternative to the *pins* hint. Additionally, this hint allows specifying pin numbers greater than 31. The numbers can be decimal or hexadecimal with 0x prefix. Any non-digit character can be used as a separator. For example, it can be a comma, a slash or a space. The separator can be followed by any number of space characters.

The following device.hints(5) are only provided by the **ar71xx_gpio** driver:

hint.gpio.%d.pinmask This is a bitmask of pins on the GPIO board that we would like to expose for use to the host operating system. To expose pin 0, 4 and 7, use the bitmask of 10010001 converted to the hexadecimal value 0x0091.

hint.gpio.%d.pinon This is a bitmask of pins on the GPIO board that will be set to ON at host start. To set pin 2, 5 and 13 to be set ON at boot, use the bitmask of 10000000010010 converted to the hexadecimal value 0x2012.

hint.gpio.function_set

hint.gpio.function_clear These are bitmasks of pins that will remap a pin to handle a specific function (USB, UART TX/RX, etc) in the Atheros function registers. This is mainly used to set/clear functions that we need when they are set up or not set up by uBoot.

Simply put, each pin of the GPIO interface is connected to an input/output of some device in a system.

SEE ALSO

gpioiic(4), gpioled(4), iicbus(4), device.hints(5), gpioctl(8)

HISTORY

The **gpiobus** manual page first appeared in FreeBSD 10.0.

AUTHORS

This manual page was written by Sean Bruno <*sbruno@FreeBSD.org*>.