Lindus Embedded

GPIO (General Purpose Input/Output) interfaces in Linux

What are GPIOs?

GPIOs are very common in embedded hardware, representing pins connected to ball grid array (BGA) packages.They are commonly used to provide flexibility in System On chip (SOC), and it is common for pins to be multiplexed and shared for different functionality. As such GPIOs are specific to each platform and we can only speak in generic terms.

Usually they can be used as writable output and readable inputs which can also be used as IRQs.

Generic GPIO API

Linux platforms that support GPIO conventions declare GENERIC_GPIO, and drivers use the interface in linux/gpio.h. GPIOs are identified by platform specific integer numbers, with negative numbers being invalid.

```
int gpio_is_valid(int number);
```

Check whether a given number is a valid GPIO.

```
int gpio_request(unsigned gpio, const char *label);
```

Request a GPIO returning 0 on success or a negative error value.

```
void gpio_free(unsigned gpio);
```

Frees a previously requested GPIO.

```
int gpio_direction_input(unsigned gpio);
int gpio_direction_output(unsigned gpio, int value);
```

Set the direction of the GPIO returning 0 on success or a negative error value.

```
int gpio_get_value(unsigned gpio);
void gpio_set_value(unsigned gpio, int value);
```

The getter return 0 or 1 (high or low), and the setter sets 0 or 1 (high or low).

Most GPIO controllers are accessed without the need for sleep and can be safely accessed through IRQ handlers. Some use message buses like I2C or SPI which may sleep, and this should be accessed by the following accessors:

```
int gpio_get_value_cansleep(unsigned gpio);
void gpio_set_value_cansleep(unsigned gpio, int value);
```

Platform code will have defined the mapping between the GPIOs and IRQs number namespaces, so we can use the following functions to swap between them.

```
int gpio_to_irq(unsigned gpio);
int irq_to_gpio(unsigned irq);
```

Not al GPIOs can be configured as IRQs and in that case they will return a negative errno.

GPIOLIB

An optional framework to support diferrent GPIO controllers (packaged as a "struct gpio_chip") with the same API. With debugfs available, **/sys/kernel/debug/gpio** lists all registered controllers and the state of used GPIOs. The platform will define ARCH_REQUIRE_GPIOLIB to compile the framework in, ARCH_WANT_OPTIONAL_GPIOLIB to let the user built it into the kernel optionally, or none if GPIOLIB is not supported.

SYSFS support

When using GPIOLIB, a sysfs interface can be enabled to control GPIO directon and value from userspace. The control interface lives in **/sys/class/gpio**.

```
echo 19 &gt; /sys/class/gpio/export
```

Will create a gpio19 node and request the GPIO number 19.

```
echo 19 > /sys/class/gpio/unexport
```

Will free the GPIO 19.

```
echo "in" > /sys/class/gpio/gpio19/direction
echo "out" > /sys/class/gpio/gpio19/direction
```

Will configure it as input or output. "low" and "out" have the same effect, to change the pin to an output which is initially low. "high" changes the pin to an output which is initially high. "in" changes the pin to an input

```
cat /sys/class/gpio/gpio19/value
```

Will read from an input GPIO.

```
echo 0 > /sys/class/gpio/gpio19/value
echo 1 > /sys/class/gpio/gpio19/value
```

Will write to an output GPIO.

```
echo rising > /sys/class/gpio/gpio19/edge
echo falling > /sys/class/gpio/gpio19/edge
echo both > /sys/class/gpio/gpio19/edge
echo none > /sys/class/gpio/gpio19/edge
```

If the GPIO can be configured as IRQ,the above configure the signal edges.

Drivers can also make GPIOs available through sysfs using:

```
int gpio_export(unsigned gpio, bool direction_may_change);
void gpio_unexport();
```

And the following can be used to symlink to a different sysfs location to provide the interface under a different device.

```
int gpio_export_link(struct device *dev, const char *name,unsigned gpio)
```

Post a comment — Trackback URI RSS 2.0 feed for these

comments This entry (permalink) was posted on Thursday,

December 29, 2011, at 4:19 pm by Alex. Filed in Uncategorized.