GPIO register access

Functions

void	**bcm2835_gpio_fsel** (uint8_t pin, uint8_t mode)
void	**bcm2835_gpio_set** (uint8_t pin)
void	**bcm2835_gpio_clr** (uint8_t pin)
void	**bcm2835_gpio_set_multi** (uint32_t mask)
void	**bcm2835_gpio_clr_multi** (uint32_t mask)
uint8_t	**bcm2835_gpio_lev** (uint8_t pin)
uint8_t	**bcm2835_gpio_eds** (uint8_t pin)
void	**bcm2835_gpio_set_eds** (uint8_t pin)
void	**bcm2835_gpio_ren** (uint8_t pin)
void	**bcm2835_gpio_clr_ren** (uint8_t pin)
void	**bcm2835_gpio_fen** (uint8_t pin)
void	**bcm2835_gpio_clr_fen** (uint8_t pin)
void	**bcm2835_gpio_hen** (uint8_t pin)
void	**bcm2835_gpio_clr_hen** (uint8_t pin)
void	**bcm2835_gpio_len** (uint8_t pin)
void	**bcm2835_gpio_clr_len** (uint8_t pin)
void	**bcm2835_gpio_aren** (uint8_t pin)
void	**bcm2835_gpio_clr_aren** (uint8_t pin)
void	**bcm2835_gpio_afen** (uint8_t pin)
void	**bcm2835_gpio_clr_afen** (uint8_t pin)
void	**bcm2835_gpio_pud** (uint8_t pud)
void	**bcm2835_gpio_pudclk** (uint8_t pin, uint8_t on)
uint32_t	**bcm2835_gpio_pad** (uint8_t group)
void	**bcm2835_gpio_set_pad** (uint8_t group, uint32_t control)
void	**bcm2835_delay** (unsigned int millis)
void	**bcm2835_delayMicroseconds** (uint64_t micros)
void	**bcm2835_gpio_write** (uint8_t pin, uint8_t on)
void	**bcm2835_gpio_write_multi** (uint32_t mask, uint8_t on)
void	**bcm2835_gpio_write_mask** (uint32_t value, uint32_t mask)
void	**bcm2835_gpio_set_pud** (uint8_t pin, uint8_t pud)

Detailed Description

These functions allow you to control the GPIO interface. You can set the function of each GPIO pin, read the input state and set the output state.

Function Documentation

void bcm2835_delay (unsigned int millis)

Delays for the specified number of milliseconds. Uses nanosleep(), and therefore does not use CPU until the time is up. However, you are at the mercy of nanosleep(). From the manual for nanosleep(): If the interval specified in req is not an exact multiple of the granularity underlying clock (see time(7)), then the interval will be rounded up to the next multiple. Furthermore, after the sleep completes, there may still be a delay before the CPU becomes free to once again execute the calling thread.

Parameters

[in] **millis** Delay in milliseconds

void bcm2835_delayMicroseconds (uint64_t micros)

Delays for the specified number of microseconds. Uses a combination of nanosleep() and a busy wait loop on the BCM2835 system timers, However, you are at the mercy of nanosleep(). From the manual for nanosleep(): If the interval specified in req is not an exact multiple of the granularity underlying clock (see time(7)), then the interval will be rounded up to the next multiple. Furthermore, after the sleep completes, there may still be a delay before the CPU becomes free to once again execute the calling thread. For times less than about 450 microseconds, uses a busy wait on the System Timer. It is reported that a delay of 0 microseconds on RaspberryPi will in fact result in a delay of about 80 microseconds. Your mileage may vary.

Parameters

[in] **micros** Delay in microseconds

void bcm2835_gpio_afen (uint8_t pin)

Enable Asynchronous Falling Edge Detect Enable for the specified pin. When a falling edge is detected, sets the appropriate pin in Event Detect Status. Asynchronous means the incoming signal is not sampled by the system clock. As such falling edges of very short duration can be detected.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_aren (uint8_t pin)

Enable Asynchronous Rising Edge Detect Enable for the specified pin. When a rising edge is detected, sets the appropriate pin in Event Detect Status. Asynchronous means the incoming signal is not sampled by the system clock. As such rising edges of very short duration can be detected.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_clr (uint8_t pin)

Sets the specified pin output to LOW.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

See Also

bcm2835_gpio_write()

void bcm2835_gpio_clr_afen (uint8_t pin)

Disable Asynchronous Falling Edge Detect Enable for the specified pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_clr_aren (uint8_t pin)

Disable Asynchronous Rising Edge Detect Enable for the specified pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_clr_fen (uint8_t pin)

Disable Falling Edge Detect Enable for the specified pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_clr_hen (uint8_t pin)

Disable High Detect Enable for the specified pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_clr_len (uint8_t pin)

Disable Low Detect Enable for the specified pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_clr_multi (uint32_t mask)

Sets any of the first 32 GPIO output pins specified in the mask to LOW.

Parameters

[in] **mask** Mask of pins to affect. Use eg: (1 << RPI_GPIO_P1_03) | (1 << RPI_GPIO_P1_05)

See Also

bcm2835_gpio_write_multi()

void bcm2835_gpio_clr_ren (uint8_t pin)

Disable Rising Edge Detect Enable for the specified pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

uint8_t bcm2835_gpio_eds (uint8_t pin)

Event Detect Status. Tests whether the specified pin has detected a level or edge as requested by **bcm2835_gpio_ren()**, **bcm2835_gpio_fen()**, **bcm2835_gpio_hen()**, **bcm2835_gpio_len()**, **bcm2835_gpio_aren()**, **bcm2835_gpio_afen()**. Clear the flag for a given pin by calling bcm2835_gpio_set_eds(pin);

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

Returns

HIGH if the event detect status for the given pin is true.

void bcm2835_gpio_fen (uint8_t pin)

Enable Falling Edge Detect Enable for the specified pin. When a falling edge is detected, sets the appropriate pin in Event Detect Status. The GPRENn registers use synchronous edge detection. This means the input signal is sampled using the system clock and then it is looking for a ?100? pattern on the sampled signal. This has the effect of suppressing glitches.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_fsel (uint8_t pin,
uint8_t mode
)

Sets the Function Select register for the given pin, which configures the pin as Input, Output or one of the 6 alternate functions.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

[in] **mode** Mode to set the pin to, one of BCM2835_GPIO_FSEL_* from **bcm2835FunctionSelect**

void bcm2835_gpio_hen (uint8_t pin)

Enable High Detect Enable for the specified pin. When a HIGH level is detected on the pin, sets the appropriate pin in Event Detect Status.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_len (uint8_t pin)

Enable Low Detect Enable for the specified pin. When a LOW level is detected on the pin, sets the appropriate pin in Event Detect Status.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

uint8_t bcm2835_gpio_lev (uint8_t pin)

Reads the current level on the specified pin and returns either HIGH or LOW. Works whether or not the pin is an input or an output.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

Returns

the current level either HIGH or LOW

uint32_t bcm2835_gpio_pad (uint8_t group)

Reads and returns the Pad Control for the given GPIO group.

Parameters

[in] **group** The GPIO pad group number, one of BCM2835_PAD_GROUP_GPIO_*

Returns

Mask of bits from BCM2835_PAD_* from **bcm2835PadGroup**

void bcm2835_gpio_pud (uint8_t pud)

Sets the Pull-up/down register for the given pin. This is used with **bcm2835_gpio_pudclk()** to set the Pull-up/down resistor for the given pin. However, it is usually more convenient to use **bcm2835_gpio_set_pud()**.

Parameters

[in] **pud** The desired Pull-up/down mode. One of BCM2835_GPIO_PUD_* from bcm2835PUDControl

See Also

bcm2835_gpio_set_pud()

void bcm2835_gpio_pudclk (uint8_t pin,
uint8_t on
)

Clocks the Pull-up/down value set earlier by **bcm2835_gpio_pud()** into the pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

[in] **on** HIGH to clock the value from **bcm2835_gpio_pud()** into the pin. LOW to remove the clock.

See Also

bcm2835_gpio_set_pud()

void bcm2835_gpio_ren (uint8_t pin)

Enable Rising Edge Detect Enable for the specified pin. When a rising edge is detected, sets the appropriate pin in Event Detect Status. The GPRENn registers use synchronous edge detection. This means the input signal is sampled using the system clock and then it is looking for a ?011? pattern on the sampled signal. This has the effect of suppressing glitches.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_set (uint8_t pin)

Sets the specified pin output to HIGH.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

See Also

bcm2835_gpio_write()

void bcm2835_gpio_set_eds (uint8_t pin)

Sets the Event Detect Status register for a given pin to 1, which has the effect of clearing the flag. Use this afer seeing an Event Detect Status on the pin.

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

void bcm2835_gpio_set_multi (uint32_t mask)

Sets any of the first 32 GPIO output pins specified in the mask to HIGH.

Parameters

[in] **mask** Mask of pins to affect. Use eg: (1 << RPI_GPIO_P1_03) | (1 << RPI_GPIO_P1_05)

See Also

bcm2835_gpio_write_multi()

void bcm2835_gpio_set_pad (uint8_t group,
uint32_t control
)

Sets the Pad Control for the given GPIO group.

Parameters

[in] **group** The GPIO pad group number, one of BCM2835_PAD_GROUP_GPIO_*

[in] **control** Mask of bits from BCM2835_PAD_* from bcm2835PadGroup. Note that it is not necessary to include BCM2835_PAD_PASSWRD in the mask as this is automatically included.

void bcm2835_gpio_set_pud (uint8_t pin,
uint8_t pud
)

Sets the Pull-up/down mode for the specified pin. This is more convenient than clocking the mode in with bcm2835_gpio_pud() and bcm2835_gpio_pudclk().

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from RPiGPIOPin.

[in] **pud** The desired Pull-up/down mode. One of BCM2835_GPIO_PUD_* from bcm2835PUDControl

```
void bcm2835_gpio_write ( uint8_t  pin,
                          uint8_t  on
                        )
```

Sets the output state of the specified pin

Parameters

[in] **pin** GPIO number, or one of RPI_GPIO_P1_* from **RPiGPIOPin**.

[in] **on** HIGH sets the output to HIGH and LOW to LOW.

```
void bcm2835_gpio_write_mask ( uint32_t  value,
                               uint32_t  mask
                             )
```

Sets the first 32 GPIO output pins specified in the mask to the value given by value

Parameters

[in] **value** values required for each bit masked in by mask, eg: (1 << RPI_GPIO_P1_03) | (1 << RPI_GPIO_P1_05)

[in] **mask** Mask of pins to affect. Use eg: (1 << RPI_GPIO_P1_03) | (1 << RPI_GPIO_P1_05)

```
void bcm2835_gpio_write_multi ( uint32_t  mask,
                                uint8_t   on
                              )
```

Sets any of the first 32 GPIO output pins specified in the mask to the state given by on

Parameters

[in] **mask** Mask of pins to affect. Use eg: (1 << RPI_GPIO_P1_03) | (1 << RPI_GPIO_P1_05)

[in] **on** HIGH sets the output to HIGH and LOW to LOW.