NAME

pwmbus, **PWMBUS_CHANNEL_CONFIG**, **PWMBUS_CHANNEL_COUNT**, **PWMBUS_CHANNEL_ENABLE**, **PWMBUS_CHANNEL_GET_CONFIG**, **PWMBUS_CHANNEL_GET_FLAGS**, **PWMBUS_CHANNEL_IS_ENABLED**, **PWMBUS_CHANNEL_SET_FLAGS**, **PWMBUS_GET_BUS** - PWMBUS methods

SYNOPSIS

device pwm
#include <pwmbus_if.h>

int
PWMBUS_CHANNEL_CONFIG(*device_t bus*, *u_int channel*, *u_int period*, *u_int duty*);

int
PWMBUS_CHANNEL_COUNT(*device_t bus*, *u_int *nchannel*);

int
PWMBUS_CHANNEL_ENABLE(*device_t bus*, *u_int channel*, *bool enable*);

int
PWMBUS_CHANNEL_GET_CONFIG(*device_t bus*, *u_int channel*, *u_int *period*, *u_int *duty*);

int
PWMBUS_CHANNEL_GET_FLAGS(*device_t bus*, *u_int channel*, *uint32_t *flags*);

int
PWMBUS_CHANNEL_IS_ENABLED(*device_t bus*, *u_int channel*, *bool *enabled*);

int
PWMBUS_CHANNEL_SET_FLAGS(*device_t bus*, *u_int channel*, *uint32_t flags*);

DESCRIPTION

The PWMBUS (Pulse-Width Modulation) interface allows a device driver to register to a global bus so other devices in the kernel can use them in a generic way.

For all **pwmbus** methods, the *period* argument is the duration in nanoseconds of one complete on-off cycle, and the *duty* argument is the duration in nanoseconds of the on portion of that cycle.

Some PWM hardware is organized as a single controller with multiple channels. Channel numbers count up from zero. When multiple channels are present, they sometimes share a common clock or other resources. In such cases, changing the period or duty cycle of any one channel may affect other channels within the hardware which share the same resources. Consult the documentation for the underlying PWM hardware device driver for details on channels that share resources.

INTERFACE

PWMBUS_CHANNEL_CONFIG(*device_t bus*, *u_int channel*, *u_int period*, *u_int duty*)

Configure the period and duty (in nanoseconds) in the PWM controller on the bus for the specified channel. Returns 0 on success or EINVAL if the values are not supported by the controller or EBUSY if the PWMBUS controller is in use and does not support changing the value on the fly.

PWMBUS_CHANNEL_COUNT(*device_t bus*, *u_int *nchannel*)

Get the number of channels supported by the controller.

PWMBUS_CHANNEL_ENABLE(*device_t bus*, *u_int channel*, *bool enable*)

Enable the PWM channel.

PWMBUS_CHANNEL_GET_CONFIG(*device_t bus*, *u_int channel*, *u_int *period*, *u_int *duty*)

Get the current configuration of the period and duty for the specified channel.

PWMBUS_CHANNEL_GET_FLAGS(*device_t bus*, *u_int channel*, *uint32_t *flags*)

Get the current flags for the channel. If the driver or controller does not support this, a default method returns a flags value of zero.

PWMBUS_CHANNEL_IS_ENABLED(*device_t bus*, *u_int channel*, *bool *enable*)

Test whether the PWM channel is enabled.

PWMBUS_CHANNEL_SET_FLAGS(*device_t bus*, *u_int channel*, *uint32_t flags*)

Set the flags of the channel (such as inverted polarity). If the driver or controller does not support this a do-nothing default method is used.

HISTORY

The **pwmbus** interface first appear in FreeBSD 13.0. The **pwmbus** interface and manual page was written by Emmanuel Vadot <*manu@FreeBSD.org*>.