PIC32 SDK

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 p32_uart Struct Reference

Public Attributes

- volatile p32_regset mode
- volatile p32_regset sta
- volatile p32_regbuf txreg
- volatile p32_regbuf rxreg
- volatile p32_regset brg

3.1.1 Member Data Documentation

3.1.1.1 brg

```
volatile p32_regset p32_uart::brg
```

3.1.1.2 mode

```
volatile p32_regset p32_uart::mode
```

3.1.1.3 rxreg

```
volatile p32_regbuf p32_uart::rxreg
```

3.1.1.4 sta

```
volatile p32_regset p32_uart::sta
```

3.1.1.5 txreg

```
volatile p32_regbuf p32_uart::txreg
```

The documentation for this struct was generated from the following file:

- drivers/uart.c

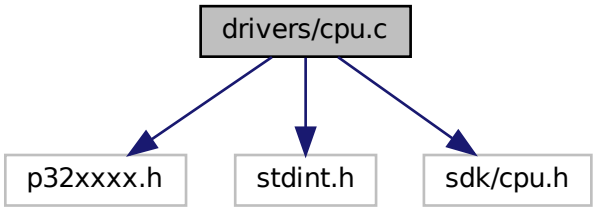
Chapter 4

File Documentation

4.1 drivers/cpu.c File Reference

```
#include <p32xxxx.h>
#include <stdint.h>
#include "sdk/cpu.h"
```

Include dependency graph for cpu.c:



Functions

- uint32_t cpu_get_peripheral_clock ()
- uint32_t cpu_get_system_clock ()
- int cpu_get_interrupt_flag (uint8_t irq)
- void cpu_set_interrupt_flag (uint8_t irq)
- void cpu_clear_interrupt_flag (uint8_t irq)
- void cpu_set_interrupt_enable (uint8_t irq)
- void cpu_clear_interrupt_enable (uint8_t irq)
- int cpu_get_interrupt_enable (uint8_t irq)
- void cpu_set_interrupt_priority (uint8_t vec, uint8_t ipl, uint8_t spl)
- void cpu_unlock ()
- void cpu_lock ()
- void cpu_reset ()

4.1.1 Function Documentation

4.1.1.1 cpu_clear_interrupt_enable()

```
void cpu_clear_interrupt_enable (
            uint8_t irq )
```

4.1.1.2 cpu_clear_interrupt_flag()

```
void cpu_clear_interrupt_flag (
            uint8_t irq )
```

4.1.1.3 cpu_get_interrupt_enable()

```
int cpu_get_interrupt_enable (
            uint8_t irq )
```

4.1.1.4 cpu_get_interrupt_flag()

```
int cpu_get_interrupt_flag (
            uint8_t irq )
```

4.1.1.5 cpu_get_peripheral_clock()

```
uint32_t cpu_get_peripheral_clock ( )
```

4.1.1.6 cpu_get_system_clock()

```
uint32_t cpu_get_system_clock ( )
```

4.1.1.7 cpu_lock()

```
void cpu_lock ( )
```

4.1.1.8 cpu_reset()

```
void cpu_reset ( )
```

4.1.1.9 cpu_set_interrupt_enable()

```
void cpu_set_interrupt_enable (
            uint8_t irq )
```

4.1.1.10 cpu_set_interrupt_flag()

```
void cpu_set_interrupt_flag (
            uint8_t irq )
```

4.1.1.11 cpu_set_interrupt_priority()

```
void cpu_set_interrupt_priority (
            uint8_t vec,
            uint8_t ipl,
            uint8_t spl )
```

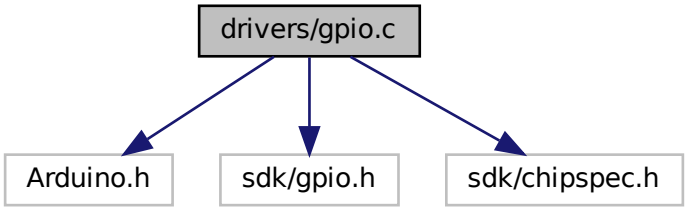
4.1.1.12 cpu_unlock()

```
void cpu_unlock ( )
```

4.2 drivers/gpio.c File Reference

```
#include <Arduino.h>
#include "sdk/gpio.h"
#include "sdk/chipspec.h"
```

Include dependency graph for gpio.c:



Classes

- struct **ppsPinMapping**

Macros

- #define NUM_PPS_PINS (sizeof(ppsPinMappingPins) / sizeof(ppsPinMappingPins[0]))
- #define NUM_PPS_FUNCTIONS (sizeof(ppsPinMappingFunctions) / sizeof(ppsPinMappingFunctions[0]))

Functions

- void gpio_set_mode (uint8_t pin, uint8_t mode)
- uint8_t gpio_read (uint8_t pin)
- void gpio_write (uint8_t pin, uint8_t val)
- int gpio_set_input_function (uint8_t pin, uint8_t function)
- int gpio_set_output_function (uint8_t pin, uint8_t function)
- int gpio_clear_output_function (uint8_t pin)
- void gpio_unlock_pps ()
- void gpio_lock_pps ()

Variables

- const struct ppsPinMapping ppsPinMappingPins [ ]
- const struct ppsPinMapping ppsPinMappingFunctions [ ]

4.2.1 Macro Definition Documentation

4.2.1.1 NUM_PPS_FUNCTIONS

```
#define NUM_PPS_FUNCTIONS (sizeof(ppsPinMappingFunctions) / sizeof(ppsPinMappingFunctions[0]))
```

4.2.1.2 NUM_PPS_PINS

```
#define NUM_PPS_PINS (sizeof(ppsPinMappingPins) / sizeof(ppsPinMappingPins[0]))
```

4.2.2 Function Documentation

4.2.2.1 gpio_clear_output_function()

```
int gpio_clear_output_function (
            uint8_t pin )
```

4.2.2.2 gpio_lock_pps()

```
void gpio_lock_pps ( )
```

4.2.2.3 gpio_read()

```
uint8_t gpio_read (
            uint8_t pin )
```

4.2.2.4 gpio_set_input_function()

```
int gpio_set_input_function (
            uint8_t pin,
            uint8_t function )
```

4.2.2.5 gpio_set_mode()

```
void gpio_set_mode (
            uint8_t pin,
            uint8_t mode )
```

4.2.2.6 gpio_set_output_function()

```
int gpio_set_output_function (
            uint8_t pin,
            uint8_t function )
```

4.2.2.7 gpio_unlock_pps()

```
void gpio_unlock_pps ( )
```

4.2.2.8 gpio_write()

```
void gpio_write (
            uint8_t pin,
            uint8_t val )
```

4.2.3 Variable Documentation

4.2.3.1 ppsPinMappingFunctions

```
const struct ppsPinMapping ppsPinMappingFunctions[]
```

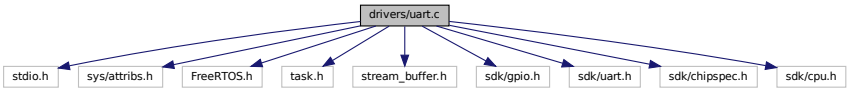
4.2.3.2 ppsPinMappingPins

```
const struct ppsPinMapping ppsPinMappingPins[]
```

4.3 drivers/uart.c File Reference

```
#include <stdio.h>
#include <sys/attribs.h>
#include "FreeRTOS.h"
#include "task.h"
#include "stream_buffer.h"
#include "sdk/gpio.h"
#include "sdk/uart.h"
#include "sdk/chipspec.h"
#include "sdk/cpu.h"
```

Include dependency graph for uart.c:



Classes

- struct p32_uart
- struct **uartControlDataStruct**

Functions

- int uart_write (uint8_t uart, uint8_t byte)
- int uart_read (uint8_t uart)
- int uart_set_tx_pin (uint8_t uart, uint8_t pin)
- int uart_set_rx_pin (uint8_t uart, uint8_t pin)
- int uart_set_baud (uint8_t uart, uint32_t baud)
- int uart_set_format (uint8_t uart, uint8_t format)
- int uart_open (uint8_t uart)
- int uart_close (uint8_t uart)

4.3.1 Function Documentation

4.3.1.1 uart_close()

```
int uart_close (
            uint8_t uart )
```

4.3.1.2 uart_open()

```
int uart_open (
             uint8_t uart )
```

4.3.1.3 uart_read()

```
int uart_read (
             uint8_t uart )
```

4.3.1.4 uart_set_baud()

```
int uart_set_baud (
             uint8_t uart,
             uint32_t baud )
```

Configure the baud rate of the selected UART

Parameters

uart	The index of the UART
baud	The baud rate to configure.

Returns

1 on success, 0 on failure

4.3.1.5 uart_set_format()

```
int uart_set_format (
             uint8_t uart,
             uint8_t format )
```

Configure the data format for the selected UART Formats are specified in convenient macros:

- uart8N1
- uart8N2
- uart8E1
- uart8E2
- uart801
- uart8O2
- uart9N1
- uart9N2

Parameters

uart	The index of the UART
format	The format to use.

Returns

1 on success, 0 on failure

4.3.1.6 uart_set_rx_pin()

```
int uart_set_rx_pin (
            uint8_t uart,
            uint8_t pin )
```

Configure the RX pin of the selected UART through PPS

Parameters

uart	The index of the UART
pin	The index of the pin to assign the RX function to

Returns

1 on success, 0 on failure

4.3.1.7 uart_set_tx_pin()

```
int uart_set_tx_pin (
            uint8_t uart,
            uint8_t pin )
```

Configure the TX pin of the selected UART through PPS

Parameters

uart	The index of the UART
pin	The index of the pin to assign the TX function to

Returns

1 on success, 0 on failure

4.3.1.8 uart_write()

```
int uart_write (
              uint8_t uart,
              uint8_t byte )
```

Index