PIC32 SDK

Generated by Doxygen 1.8.17

1 Data Structure Index 1

1.1 Data Structures . . . 1

2 File Index 3

2.1 File List . . . 3

3 Data Structure Documentation 5

3.1 p32_uart Struct Reference . . . 5

4 File Documentation 7

4.1 drivers/cpu.c File Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Function Documentation . . . 8

4.1.2.1 cpu_clear_interrupt_enable() . . . 8

4.1.2.2 cpu_clear_interrupt_flag() . . . 8

4.1.2.3 cpu_ct_init() . . . 8

4.1.2.4 cpu_get_interrupt_enable() . . . 9

4.1.2.5 cpu_get_interrupt_flag() . . . 9

4.1.2.6 cpu_get_peripheral_clock() . . . 9

4.1.2.7 cpu_get_system_clock() . . . 10

4.1.2.8 cpu_lock() . . . 10

4.1.2.9 cpu_reset() . . . 10

4.1.2.10 cpu_set_interrupt_enable() . . . 10

4.1.2.11 cpu_set_interrupt_flag() . . . 10

4.1.2.12 cpu_set_interrupt_priority() . . . 11

4.1.2.13 cpu_unlock() . . . 11

4.2 drivers/flash.c File Reference . . . 11

4.2.1 Detailed Description . . . 12

4.2.2 Function Documentation . . . 12

4.2.2.1 flash_clear_error() . . . 12

4.2.2.2 flash_erase_page() . . . 12

4.2.2.3 flash_erase_progmem() . . . 13

4.2.2.4 flash_is_ok() . . . 13

4.2.2.5 flash_lock() . . . 13

4.2.2.6 flash_operation() . . . 13

4.2.2.7 flash_unlock() . . . 14

4.2.2.8 flash_write_row() . . . 14

4.2.2.9 flash_write_word() . . . 14

4.3 drivers/gpio.c File Reference . . . 15

4.3.1 Detailed Description . . . 15

4.3.2 Function Documentation . . . 16

4.3.2.1 gpio_clear_output_function() . . . 16

4.3.2.2 gpio_connect_change_interrupt() . . . 16

4.3.2.3 gpio_connect_external_interrupt() . . . 17
4.3.2.4 gpio_disconnect_change_interrupt() . . . 17
4.3.2.5 gpio_disconnect_external_interrupt() . . . 18
4.3.2.6 gpio_read() . . . 18
4.3.2.7 gpio_set_input_function() . . . 18
4.3.2.8 gpio_set_mode() . . . 19
4.3.2.9 gpio_set_output_function() . . . 19
4.3.2.10 gpio_write() . . . 20
4.4 drivers/uart.c File Reference . . . 20
4.4.1 Detailed Description . . . 21
4.4.2 Function Documentation . . . 21
4.4.2.1 uart_close() . . . 21
4.4.2.2 uart_flush() . . . 21
4.4.2.3 uart_open() . . . 22
4.4.2.4 uart_peek() . . . 22
4.4.2.5 uart_purge() . . . 22
4.4.2.6 uart_read() . . . 23
4.4.2.7 uart_rx_available() . . . 23
4.4.2.8 uart_set_baud() . . . 23
4.4.2.9 uart_set_format() . . . 24
4.4.2.10 uart_set_rx_pin() . . . 24
4.4.2.11 uart_set_tx_pin() . . . 25
4.4.2.12 uart_tx_available() . . . 25
4.4.2.13 uart_write() . . . 25
4.4.2.14 uart_write_bytes() . . . 26

Index **27**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

p32_uart 5

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

drivers/cpu.c 7
drivers/flash.c 11
drivers/gpio.c 15
drivers/uart.c 20

Chapter 3

Data Structure Documentation

3.1 p32_uart Struct Reference

Data Fields

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

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

- drivers/uart.c

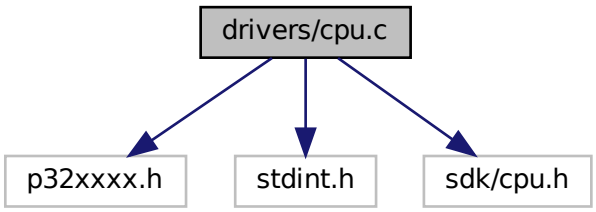
Chapter 4

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 ()
- void cpu_ct_init (uint32_t initcompare)

4.1.1 Detailed Description

Controls the internals of the CPU, interrupts, clocks, etc.

4.1.2 Function Documentation

4.1.2.1 cpu_clear_interrupt_enable()

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

Disable an interrupt

Parameters

irq	The vector (MZ) or IRQ (MX) of the interrupt to disable

4.1.2.2 cpu_clear_interrupt_flag()

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

Clear an interrupt flag. This should normally be performed in the interrupt service handler for an interrupt.

Parameters

irq	The vector (MZ) or IRQ (MX) of the interrupt to clear the flag for

4.1.2.3 cpu_ct_init()

```
void cpu_ct_init (
            uint32_t initcompare )
```

Initialize the Core Timer to zero and set an initial compare

Parameters

initcompare	Initial value to use in the Compare register

4.1.2.4 cpu_get_interrupt_enable()

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

Query if an interrupt is enabled or not

Parameters

irq	The vector (MZ) or IRQ (MX) of the interrupt to query

Returns

1 if the interrupt is enabled, 0 otherwise

4.1.2.5 cpu_get_interrupt_flag()

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

Tests if an interrupt flag is set or not

Parameters

irq	The vector (MZ) or IRQ (MX) of the interrupt you are querying

Returns

1 if the flag is set, otherwise 0.

4.1.2.6 cpu_get_peripheral_clock()

```
uint32_t cpu_get_peripheral_clock ( )
```

Calculates and returns the current peripheral bus clock frequency

Returns

The frequency in Hz

4.1.2.7 cpu_get_system_clock()

```
uint32_t cpu_get_system_clock ( )
```

Calculates and returns the current CPU clock frequency

Returns

The frequency in Hz

4.1.2.8 cpu_lock()

```
void cpu_lock ( )
```

Lock protected functions of the CPU

4.1.2.9 cpu_reset()

```
void cpu_reset ( )
```

Perform a software reset of the CPU

4.1.2.10 cpu_set_interrupt_enable()

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

Enable an interrupt

Parameters

irq	The vector (MZ) or IRQ (MX) of the interrupt to enable

4.1.2.11 cpu_set_interrupt_flag()

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

Set an interrupt flag. This has the effect of immediately causing the interrupt service handler to be called.

Parameters

irq	The vector (MZ) or IRQ (MX) of the interrupt to set the flag for

4.1.2.12 cpu_set_interrupt_priority()

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

Set the priority and sub-priority for an interrupt

Parameters

irq	The vector of the interrupt to set the priority of
ipl	The priority (0-7) for the interrupt
spl	The sub-priority (0-3) for the interrupt

4.1.2.13 cpu_unlock()

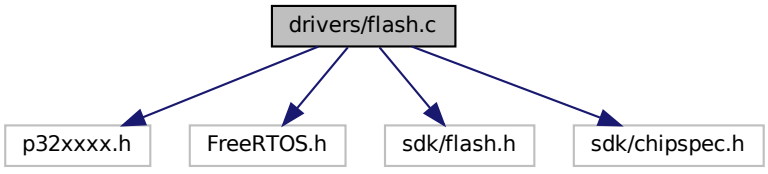
```
void cpu_unlock ( )
```

Unlock protected functions of the CPU

4.2 drivers/flash.c File Reference

```
#include <p32xxxx.h>
#include "FreeRTOS.h"
#include "sdk/flash.h"
#include "sdk/chipspec.h"
```

Include dependency graph for flash.c:



Functions

- void flash_unlock ()
- void flash_lock ()
- int flash_is_ok ()
- int flash_operation (uint32_t op)
- int flash_erase_page (void ∗adr)
- int flash_write_word (void ∗adr, uint32_t val)
- int flash_write_row (void ∗adr, void ∗val)
- int flash_clear_error ()
- int flash_erase_progmem ()

4.2.1 Detailed Description

Provides facilities for reading and writing the internal flash of the PIC32

4.2.2 Function Documentation

4.2.2.1 flash_clear_error()

```
int flash_clear_error ( )
```

Clear an error from the previous flash operation

Returns

1 if the operation succeeded, 0 otherwise

4.2.2.2 flash_erase_page()

```
int flash_erase_page (
            void * adr )
```

Erase a page of flash memory

Parameters

addr	The address of the page to erase

Returns

1 if the operation succeeded, 0 otherwise

4.2.2.3 flash_erase_progmem()

```
int flash_erase_progmem ( )
```

Erase the entirety of the program flash memory. This should only be executed from the boot segment

Returns

1 if the operation succeeded, 0 otherwise

4.2.2.4 flash_is_ok()

```
int flash_is_ok ( )
```

Test the status of the previous flash operation

Returns

1 if the operation succeeded, 0 otherwise

4.2.2.5 flash_lock()

```
void flash_lock ( )
```

Lock the flash controller. This should be done after use.

4.2.2.6 flash_operation()

```
int flash_operation (
            uint32_t op )
```

Perform a flash operation. The relevent flash registers must be set up first.

Parameters

op	The operation to perform. One of: • flashOP_NOP • flashOP_WRITE_WORD • flashOP_WRITE_ROW • flashOP_ERASE_PAGE • flashOP_ERASE_PROGMEM

Returns

1 if the operation succeeded, 0 otherwise

4.2.2.7 flash_unlock()

```
void flash_unlock ( )
```

Unlock the flash controller. This must be done before any flash operations are attempted.

4.2.2.8 flash_write_row()

```
int flash_write_row (
            void * adr,
            void * val )
```

Write a row of data into flash memory. The memory must already have been erased

Parameters

adr	The start address of the row to write to
val	Pointer to the data to write into the row

Returns

1 if the operation succeeded, 0 otherwise

4.2.2.9 flash_write_word()

```
int flash_write_word (
            void * adr,
            uint32_t val )
```

Write a single word of data into flash memory. The memory must already have been erased

Parameters

adr	The destination address to write to
val	The 32-bit word to write

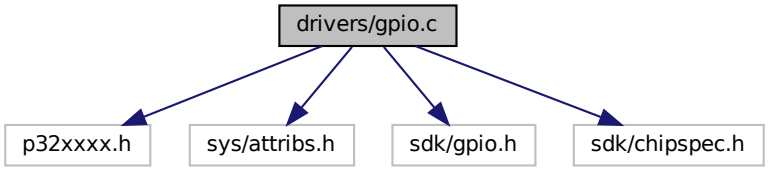
Returns

1 if the operation succeeded, 0 otherwise

4.3 drivers/gpio.c File Reference

```
#include <p32xxxx.h>
#include <sys/attribs.h>
#include "sdk/gpio.h"
#include "sdk/chipspec.h"
```

Include dependency graph for gpio.c:



Data Structures

- struct **cnInterruptCallback**
- 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)
- int gpio_connect_change_interrupt (uint8_t pin, uint8_t type, gpioISR_t callback)
- int gpio_disconnect_change_interrupt (uint8_t pin, uint8_t type)
- int gpio_connect_external_interrupt (uint8_t pin, uint8_t interrupt, uint8_t mode, void(∗callback)())
- int gpio_disconnect_external_interrupt (uint8_t interrupt)
- void **__ISR** (_EXTERNAL_0_VECTOR, IPL6)
- void **__ISR** (_EXTERNAL_1_VECTOR, IPL6)
- void **__ISR** (_EXTERNAL_2_VECTOR, IPL6)
- void **__ISR** (_EXTERNAL_3_VECTOR, IPL6)
- void **__ISR** (_EXTERNAL_4_VECTOR, IPL6)

4.3.1 Detailed Description

Routines for controlling GPIO pins in digital modes. This also includes external and change notification interrupts.

4.3.2 Function Documentation

4.3.2.1 gpio_clear_output_function()

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

Remove the output Peripheral Pin Select function from a pin returning it to normal GPIO usage

Parameters

pin	The pin to configur

Returns

1 if the function could be removed, 0 otherwise

4.3.2.2 gpio_connect_change_interrupt()

```
int gpio_connect_change_interrupt (
            uint8_t pin,
            uint8_t type,
            gpioISR_t callback )
```

Connect the Change Notfication interrupt of a GPIO pin to a callback routine. Different callback routines can be connected to both the rising edge and falling edge interrupts.

Parameters

pin	The pin to attach the callback to
type	The edge to connect to. One of: • gpioINTERRUPT_RISING • gpioINTERRUPT_FALLING
callback	The function to execute when the interrupt triggers. Two parameters passed (both uint8_t). The first is the pin number the interrupt occurred on, the second is the pin state 0 or 1.

Returns

1 if the interrupt could be configured and connected, 0 otherwise

4.3.2.3 gpio_connect_external_interrupt()

```
int gpio_connect_external_interrupt (
            uint8_t pin,
            uint8_t interrupt,
            uint8_t mode,
            void(*)() callback )
```

Connect an interrupt routine to an external interrupt. Unlike Change Notification interrupts only one edge of an external interrupt can be triggered on. However they are more responsive that change notification.

Parameters

pin	The pin to configure the interrupt on through PPS (if needed)
interrupt	The interrupt number (0-4) to connect to
mode	The edge to connect to. One of: • gpioINTERRUPT_RISING • gpioINTERRUPT_FALLING
callback	The callback to connect with the interrupt. void callback()

Returns

1 if the interrupt could be configured and connected, 0 otherwise

4.3.2.4 gpio_disconnect_change_interrupt()

```
int gpio_disconnect_change_interrupt (
            uint8_t pin,
            uint8_t type )
```

Disconnect a Change Notification callback from a pin

Parameters

pin	The pin index to disconnect the callback from
type	The edge to disconnect from. One of: • gpioINTERRUPT_RISING • gpioINTERRUPT_FALLING

Returns

1 if the interrupt could be disconnected, 0 otherwise

4.3.2.5 gpio_disconnect_external_interrupt()

```
int gpio_disconnect_external_interrupt (
            uint8_t interrupt )
```

Disconnect an external interrupt callback from a pin

Parameters

pin	The pin index to disconnect

Returns

1 if the interrupt could be disconnected, 0 otherwise.

4.3.2.6 gpio_read()

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

Read the state of a GPIO pin

Parameters

pin	The pin index to read

Returns

1 if the input is HIGH or 0 if LOW

4.3.2.7 gpio_set_input_function()

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

Configure the input Peripheral Pin Select function on a pin

Parameters

pin	The pin to configure
function	The PPS function to assign in the form gpioPPS_<FUNC>

Returns

1 if the function could be assigned, 0 otherwise

4.3.2.8 gpio_set_mode()

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

Set the IO mode for a GPIO pin

Configures a pin to be either input or output, and controls extra facilities such as open-drain and pullup/down resistors.

A pin mode is specified as a bitmap of a combination of macros to build the completed mode. One of gpioM↩ ODE_OUTPUT or gpioMODE_INPUT must be specified. for gpioMODE_OUTPUT you may also combine it with gpioMODE_OPENDRAIN. For gpioMODE_INPUT you may also combine it with gpioMODE_PULLUP and gpioM↩ ODE_PULLDOWN.

Parameters

pin	The GPIO pin index
mode	The IO mode to set

4.3.2.9 gpio_set_output_function()

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

Configure the output Peripheral Pin Select function on a pin

Parameters

pin	The pin to configure
function	The PPS function to assign in the form gpioPPS_<FUNC>

Returns

1 if the function could be assigned, 0 otherwise

4.3.2.10 gpio_write()

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

Set the output level of a GPIO pin

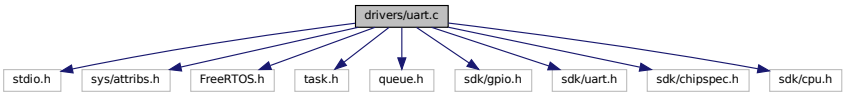
Parameters

pin	The pin index to write
val	1 to drive the pin HIGH or 0 to drive it LOW

4.4 drivers/uart.c File Reference

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

Include dependency graph for uart.c:



Data Structures

- struct p32_uart
- struct **uartControlDataStruct**

Macros

- #define **QUEUES** NULL

Functions

- void uart_purge (uint8_t uart)
- void uart_flush (uint8_t uart)
- int uart_rx_available (uint8_t uart)
- int uart_tx_available (uint8_t uart)

- int uart_write_bytes (uint8_t uart, const uint8_t *bytes, size_t len)
- int uart_write (uint8_t uart, uart_queue_t byte)
- int uart_read (uint8_t uart)
- int uart_peek (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)

4.4.1 Detailed Description

Controls the USART peripherals of the PIC32

4.4.2 Function Documentation

4.4.2.1 uart_close()

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

Close a UART and free any resources used

Parameters

uart	The UART index (0-5) to close

Returns

1 if the UART could be closed, 0 otherwise

4.4.2.2 uart_flush()

```
void uart_flush (
            uint8_t uart )
```

Block until all data is sent out of a UART transmission queue

Parameters

uart	The UART number (0-5) to flush

4.4.2.3 uart_open()

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

Open a UART. This configures the UART and starts the transmit and receive queues and interrupts.

Parameters

uart	The UART index (0-5) to open

Returns

1 if the UART could be opened, 0 otherwise

4.4.2.4 uart_peek()

```
int uart_peek (
            uint8_t uart )
```

Read the next byte from the UART receive queue without removing it from the queue param uart The number of the UART (0-5) to read from

Returns

The next byte in the queue, or -1 if the queue is empty

4.4.2.5 uart_purge()

```
void uart_purge (
            uint8_t uart )
```

Purge all received data from the receive queue of a UART

Parameters

uart	The UART number (0-5) to purge

4.4.2.6 uart_read()

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

Read a single byte from the UART receive queue

Parameters

uart	The number of the UART (0-5) to read from

Returns

The next byte in the queue, or -1 if the queue is empty

4.4.2.7 uart_rx_available()

```
int uart_rx_available (
            uint8_t uart )
```

Return the number of bytes available to read in the UART receive buffer

Parameters

uart	The UART (0-5) to query

Returns

The number of bytes waiting to be read

4.4.2.8 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

4.4.2.9 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

4.4.2.10 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

4.4.2.11 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

4.4.2.12 uart_tx_available()

```
int uart_tx_available (
            uint8_t uart )
```

Return the number of spaces left in the transmission buffer of a UART

Parameters

uart	The UART (0-5) to query

Returns

The number of bytes space in the transmit queue

4.4.2.13 uart_write()

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

Write a single byte through the UART

Parameters

uart	The UART (0-5) to write to
byte	The data to write

Returns

1 if the data could be written, 0 otherwise

4.4.2.14 uart_write_bytes()

```
int uart_write_bytes (
            uint8_t uart,
            const uint8_t * bytes,
            size_t len )
```

Write a block of bytes out through the UART

Parameters

uart	The UART (0-5) to write to
bytes	Pointer to the data to write
len	Length of the data to write

Returns

The number of bytes able to be written

Index

cpu.c
    cpu_clear_interrupt_enable, 8
    cpu_clear_interrupt_flag, 8
    cpu_ct_init, 8
    cpu_get_interrupt_enable, 8
    cpu_get_interrupt_flag, 9
    cpu_get_peripheral_clock, 9
    cpu_get_system_clock, 9
    cpu_lock, 10
    cpu_reset, 10
    cpu_set_interrupt_enable, 10
    cpu_set_interrupt_flag, 10
    cpu_set_interrupt_priority, 11
    cpu_unlock, 11
cpu_clear_interrupt_enable
    cpu.c, 8
cpu_clear_interrupt_flag
    cpu.c, 8
cpu_ct_init
    cpu.c, 8
cpu_get_interrupt_enable
    cpu.c, 8
cpu_get_interrupt_flag
    cpu.c, 9
cpu_get_peripheral_clock
    cpu.c, 9
cpu_get_system_clock
    cpu.c, 9
cpu_lock
    cpu.c, 10
cpu_reset
    cpu.c, 10
cpu_set_interrupt_enable
    cpu.c, 10
cpu_set_interrupt_flag
    cpu.c, 10
cpu_set_interrupt_priority
    cpu.c, 11
cpu_unlock
    cpu.c, 11

drivers/cpu.c, 7
drivers/flash.c, 11
drivers/gpio.c, 15
drivers/uart.c, 20

flash.c
    flash_clear_error, 12
    flash_erase_page, 12
    flash_erase_progmem, 12
    flash_is_ok, 13
    flash_lock, 13
    flash_operation, 13
    flash_unlock, 14
    flash_write_row, 14
    flash_write_word, 14
flash_clear_error
    flash.c, 12
flash_erase_page
    flash.c, 12
flash_erase_progmem
    flash.c, 12
flash_is_ok
    flash.c, 13
flash_lock
    flash.c, 13
flash_operation
    flash.c, 13
flash_unlock
    flash.c, 14
flash_write_row
    flash.c, 14
flash_write_word
    flash.c, 14

gpio.c
    gpio_clear_output_function, 16
    gpio_connect_change_interrupt, 16
    gpio_connect_external_interrupt, 16
    gpio_disconnect_change_interrupt, 17
    gpio_disconnect_external_interrupt, 17
    gpio_read, 18
    gpio_set_input_function, 18
    gpio_set_mode, 19
    gpio_set_output_function, 19
    gpio_write, 19
gpio_clear_output_function
    gpio.c, 16
gpio_connect_change_interrupt
    gpio.c, 16
gpio_connect_external_interrupt
    gpio.c, 16
gpio_disconnect_change_interrupt
    gpio.c, 17
gpio_disconnect_external_interrupt
    gpio.c, 17
gpio_read
    gpio.c, 18
gpio_set_input_function
    gpio.c, 18

gpio_set_mode
    gpio.c, 19
gpio_set_output_function
    gpio.c, 19
gpio_write
    gpio.c, 19

p32_uart, 5

uart.c
    uart_close, 21
    uart_flush, 21
    uart_open, 22
    uart_peek, 22
    uart_purge, 22
    uart_read, 22
    uart_rx_available, 23
    uart_set_baud, 23
    uart_set_format, 24
    uart_set_rx_pin, 24
    uart_set_tx_pin, 25
    uart_tx_available, 25
    uart_write, 25
    uart_write_bytes, 26
uart_close
    uart.c, 21
uart_flush
    uart.c, 21
uart_open
    uart.c, 22
uart_peek
    uart.c, 22
uart_purge
    uart.c, 22
uart_read
    uart.c, 22
uart_rx_available
    uart.c, 23
uart_set_baud
    uart.c, 23
uart_set_format
    uart.c, 24
uart_set_rx_pin
    uart.c, 24
uart_set_tx_pin
    uart.c, 25
uart_tx_available
    uart.c, 25
uart_write
    uart.c, 25
uart_write_bytes
    uart.c, 26