# EV3-OSEK

1.0

Generated by Doxygen 1.8.10

Sat May 28 2016 15:22:22

# Contents

# Chapter 1

# Data Structure Index

## 1.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

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

# Chapter 3

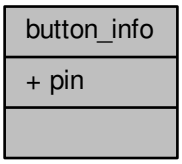
# Data Structure Documentation

## 3.1 button_info Struct Reference

This struct contains the information required to read a buttons state.

`#include <button.h>`

Collaboration diagram for button_info:




### Data Fields

- unsigned int pin

  *The pin representing the buttons state.*

### 3.1.1 Detailed Description

This struct contains the information required to read a buttons state.

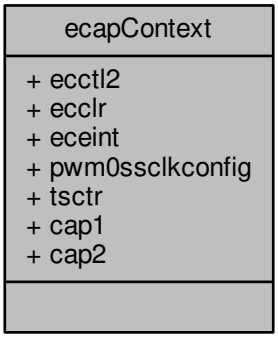
Every buttons is red by reading the signal on 1 GPIO pin which is configured as input.

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

- leJOS_EV3/src/ev3/ninja/button.h

## 3.2 ecapContext Struct Reference

Collaboration diagram for ecapContext:




### Data Fields

- unsigned short **ecctl2**
- unsigned short **ecclr**
- unsigned short **eceint**
- unsigned int **pwm0ssclkconfig**
- unsigned int **tsctr**
- unsigned int **cap1**
- unsigned int **cap2**

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

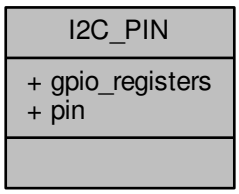
- leJOS_EV3/src/ev3/include/ecap.h

## 3.3 I2C_PIN Struct Reference

This struct represents a GPIO pin used for I2C communication.

`#include <i2c.h>`

Collaboration diagram for I2C_PIN:




### Data Fields

- U8 gpio_registers

  *The GPIO register to control this pin.*

- U8 pin

  *The number of the GPIO pin on its bank.*

### 3.3.1 Detailed Description

This struct represents a GPIO pin used for I2C communication.

Every pin is represented by a pin number on a GPIO bank (0 to 15) and one GPIO register to control that pin. Note: 2 GPIO banks always share one GPIO register, therefore bank 0 and 1 relate to register 0, bank 2 and 3 relate to register 1 and so on.

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

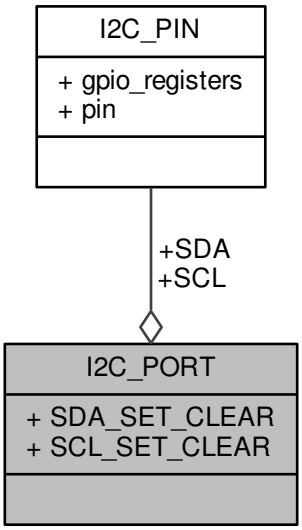
- leJOS_EV3/src/ev3/i2c.h

## 3.4 I2C_PORT Struct Reference

This struct represents an I2C port.

```
#include <i2c.h>
```

Collaboration diagram for I2C_PORT:




### Data Fields

- I2C_PIN SDA

  *GPIO pin SDA (data)*
- I2C_PIN SCL

  *GPIO pin SCL (clock)*
- unsigned int SDA_SET_CLEAR

  *Bitmask to set/clear the SDA pin.*
- unsigned int SCL_SET_CLEAR

  *Bitmask to set/clear the SCL pin.*

#### 3.4.1 Detailed Description

This struct represents an I2C port.

Every port is represented by two GPIO pins (SDA for data and SCL for clock). This struct also contains the corresponding bitmasks in order to manipulate said GPIO pins.

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

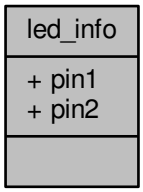
- leJOS_EV3/src/ev3/i2c.h

## 3.5 led_info Struct Reference

This struct stores information about one LED which is controlled by 2 GPIO pins.

`#include <led.h>`

Collaboration diagram for led_info:




### Data Fields

- unsigned int pin1

  *The first GPIO pin, responsible for the red LED.*
- unsigned int pin2

  *The second GPIO pin, responsible for the green LED.*

### 3.5.1 Detailed Description

This struct stores information about one LED which is controlled by 2 GPIO pins.

One GPIO pin controlls the red LED and one pin the green LED.

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

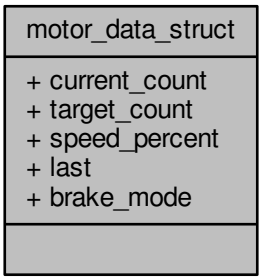
- leJOS_EV3/src/ev3/ninja/led.h

## 3.6 motor_data_struct Struct Reference

Contains the information about wheel revolution, given speed and brake mode of motors.

`#include <motor.h>`

Collaboration diagram for motor_data_struct:



### Data Fields

- int current_count

  *current rotation angle in degrees*
- int target_count

  *target rotation angle to reach*
- int speed_percent

  *not used. for future use*
- U32 last

  *last value of INT (pin5)*
- short brake_mode

  *true - brake immediately, false - float brake (not immediately).*

#### 3.6.1 Detailed Description

Contains the information about wheel revolution, given speed and brake mode of motors.

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

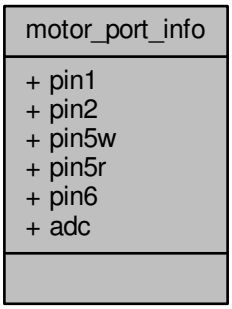
- leJOS_EV3/src/ev3/ninja/motor.h

## 3.7 motor_port_info Struct Reference

Contains the gpio pins of output and input pins.

`#include <motor.h>`

Collaboration diagram for motor_port_info:




### Data Fields

- unsigned int pin1

  *Is used by pwm module to give power to motors.*
- unsigned int pin2

  *Is used by pwm module to give power to motors.*
- unsigned int **pin5w**

- unsigned int pin5r

  *INT value. Is used to track a wheel revolution of motors.*
- unsigned int pin6

  *DIR value. Is used to track a wheel revolution of motors.*
- unsigned short **adc**

### 3.7.1 Detailed Description

Contains the gpio pins of output and input pins.

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

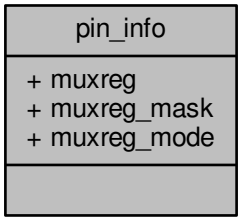
- leJOS_EV3/src/ev3/ninja/motor.h

## 3.8 pin_info Struct Reference

This struct represents 1 GPIO pin.

`#include <pininfo.h>`

Collaboration diagram for pin_info:




### Data Fields

- unsigned int muxreg

  *The PINMUX register for this pin (ranging from 0 to 19)*
- unsigned int muxreg_mask

  *The bitmask to clear the half-byte in the PINMUX register for this pin.*
- unsigned int muxreg_mode

  *The bitmask to set the half-byte in the PINMUX register for this pin to the desired value.*

### 3.8.1 Detailed Description

This struct represents 1 GPIO pin.

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

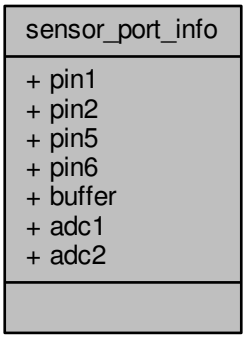
- leJOS_EV3/src/ev3/ninja/pininfo.h

## 3.9 sensor_port_info Struct Reference

This struct combines information about a sensor port of the EV3.

`#include <init.h>`

Collaboration diagram for sensor_port_info:




### Data Fields

- unsigned int pin1

  *I_ONB - 9V enable (high)*
- unsigned int pin2

  *LEGDETB - Digital input pulled up.*
- unsigned int pin5

  *DIGIB0 - Digital input/output.*
- unsigned int pin6

  *DIGIB1 - Digital input/output.*
- unsigned int buffer

  *Buffer disable.*
- unsigned char adc1

  *ADC channel 1.*
- unsigned char adc2

  *ADC channel 2.*

### 3.9.1 Detailed Description

This struct combines information about a sensor port of the EV3.

It is based on code taken from the ev3ninja project. Each sensor port is described by 4 GPIO pins which connect to the sensor, 1 buffer GPIO pin and 2 ADC channels.

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

- leJOS_EV3/src/ev3/init.h

# Chapter 4

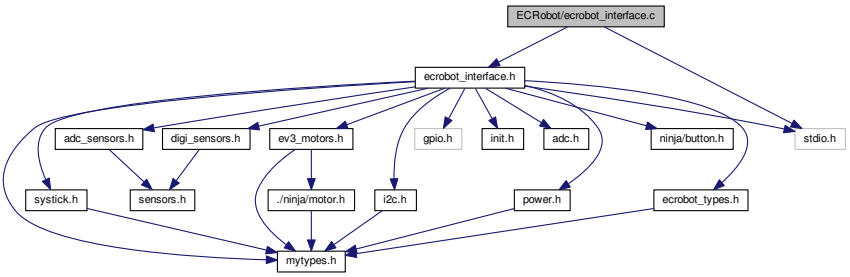
# File Documentation

## 4.1 ECRobot/ecrobot_interface.c File Reference

This file contains the function definitions for the users ECRobot-API. This API is used to interact with sensors and actuators.

```
#include "ecrobot_interface.h"
#include "stdio.h"
```

Include dependency graph for ecrobot_interface.c:




### Macros

- #define CHECK_INIT if(!isInitialized){leJOS_init();isInitialized=1;}

  *A macro that is internally used to initialize leJOS if needed.*

### Functions

- void error_function_call_not_implemented (char *function_name)

  *The internal standard error function for unimplemented ECRobot functions.*
- void ecrobot_set_motor_speed (U8 port_id, S8 speed)

  *Sets Servo Motor PWM value. Wrapper of nxt_motor_set_speed, but brake mode is fixed as brake.*
- void ecrobot_set_motor_mode_speed (U8 port_id, S32 mode, S8 speed)

  *Sets Servo Motor brake mode and PWM value. Wrapper of nxt_motor_set_speed.*
- S32 ecrobot_get_motor_rev (U8 port_id)

  *Gets Servo Motor revolution value in degree. Wrapper of nxt_motor_get_count.*
- void ecrobot_set_motor_rev (U8 port_id, S32 rev)

*Sets Servo Motor revolution value in degree. Wrapper of nxt_motor_set_count.*

- U16 ecrobot_get_light_sensor (U8 port_id)

  *Get NXT Light Sensor raw A/D data.*
- void ecrobot_set_light_sensor_active (U8 port_id)

  *Turn infra-red light on.*
- void ecrobot_set_light_sensor_inactive (U8 port_id)

  *Turn infra-red light off.*
- U8 ecrobot_get_touch_sensor (U8 port_id)

  *Get Touch Sensor on/off status.*
- U16 ecrobot_get_sound_sensor (U8 port_id)

  *Get Sound Sensor raw A/D data.*
- void ecrobot_init_i2c (U8 port_id, U8 type)

  *Init a NXT sensor port for I2C communication.*
- U8 ecrobot_wait_i2c_ready (U8 port_id, U32 wait)

  *Wait until I2C communication is ready.*
- SINT ecrobot_send_i2c (U8 port_id, U32 address, SINT i2c_reg, U8 *buf, U32 len)

  *Send I2C data.*
- SINT ecrobot_read_i2c (U8 port_id, U32 address, SINT i2c_reg, U8 *buf, U32 len)

  *Read I2C data.*
- void ecrobot_term_i2c (U8 port_id)

  *Terminate a NXT sensor port used for I2C communication.*
- void ecrobot_init_sonar_sensor (U8 port_id)

  *Init a NXT sensor port for Ultrasonic Sensor.*
- S32 ecrobot_get_sonar_sensor (U8 port_id)

  *Get Ultrasonic Sensor measurement data in cm.*
- void ecrobot_get_sonar_sensor_single_shot (U8 port_id, U8 data_buffer[8])

  *Set the mode of the Lego ultrasonic sensor at the specified port to ULTRASONIC_MODE_SINGLE_SHOT. After that get the range of the Lego ultrasonic sensor connected at the specified port and store it in the buffer.*
- void ecrobot_term_sonar_sensor (U8 port_id)

  *Terminate I2C used for for Ultrasonic.*
- U16 ecrobot_get_gyro_sensor (U8 port_id)

  *Gets Gyro Sensor raw A/D data. The sensor data has offset value (approximately 600).*
- S16 ecrobot_get_gyro_sensor_degrees (U8 port_id)

  *Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.*
- void ecrobot_init_accel_sensor (U8 port_id)

  *Initializes a port for I2C communication for Acceleration Sensor. This function should be implemented in the device initialize hook routine.*
- void ecrobot_get_accel_sensor (U8 port_id, S16 buf[3])

  *Gets acceleration data in three axes. The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will call sensor_accel_calibrate when called for the first time. The values returned will be relativ to 0.*
- void ecrobot_term_accel_sensor (U8 port_id)

  *Terminates I2C communication used for Acceleraton Sensor. This function should be implemented in the device terminate hook routine.*
- void ecrobot_init_color_sensor (U8 port_id)

  *initializes a port for I2C communication for the Color Sensor. This function should be implemented in the device initialize hook routine.*
- U8 ecrobot_cal_color_sensor (U8 port_id, U8 mode)

  *Calibrate the HiTechnic color sensor at the specified port.*
- void ecrobot_get_color_sensor (U8 port_id, S16 buf[3])

  *Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.*
- void ecrobot_term_color_sensor (U8 port_id)

*Terminates I2C communication used for the Color Sensor. This function should be implemented in the device terminate hook routine.*

- void ecrobot_init_compass_sensor (U8 port_id)

  *Initializes a port for I2C communication for Compass Sensor. This function should be implemented in the device initialize hook routine.*
- S16 ecrobot_get_compass_sensor (U8 port_id)

  *Read the current direction meassured by the HiTechnic compass sensor at the specified port.*
- void ecrobot_term_compass_sensor (U8 port_id)

  *Terminates I2C communication used for Compass Sensor. This function should be implemented in the device terminate hook routine.*
- void ecrobot_cal_start_compass_sensor (U8 port_id)

  *Start the HiTechnic compass sensor calibration.*
- U8 ecrobot_cal_end_compass_sensor (U8 port_id)

  *Finish the HiTechnic compass sensor calibration.*
- void ecrobot_init_temperature_sensor (U8 port_id)

  *Initializes a port for I2C communication for Temperature Sensor.*
- float ecrobot_get_temperature_sensor (U8 port_id)

  *Measures and returns the current temperature value.*
- void ecrobot_setResolution_temperature_sensor (U8 port_id, U8 resolution)

  *Set the resolution for the measured temperature values.*
- U8 ecrobot_getResolution_temperature_sensor (U8 port_id)

  *Returns the current resolution setting of the sensor.*
- void ecrobot_term_temperature_sensor (U8 port_id)

  *Terminates I2C communication used for Temperature Sensor.*
- U16 ecrobot_get_battery_voltage (void)

  *Get the voltage of the battery.*
- U16 ecrobot_get_battery_current (void)

  *Get the current of the battery.*
- U32 ecrobot_get_systick_ms (void)

  *Get the current tick in milliseconds.*
- void ecrobot_wait_ms (U32 ms)

  *Wait for the specified amount of time.*
- U8 ecrobot_is_ENTER_button_pressed (void)

  *Returns status of the enter button.*
- U8 ecrobot_is_RUN_button_pressed (void)

  *Returns status of the run button.*

## Variables

- unsigned char isInitialized = 0

  *A internally used attribute. Needed to save the init state.*
- sensor_port_info ports [ ]

  *A internally used attribute. Needed for port access.*

### 4.1.1 Detailed Description

This file contains the function definitions for the users ECRobot-API. This API is used to interact with sensors and actuators.

**Author**

Christian Soward, Tobias Schießl

### 4.1.2 Function Documentation

#### 4.1.2.1 U8 ecrobot_cal_color_sensor ( U8 *port_id,* U8 *mode* )

Calibrate the HiTechnic color sensor at the specified port.

To calibrate the sensor properly, this function has to be called two times. Once with mode set to CAL_WHITE (0x043) and once with mode set to CAL_BLACK (0x42). Calibration information is stored directly on the sensor and will be persistent, even if the sensor is no longer provided with power. When called with mode set to CAL_WHITE, the sensor should be located in front of a diffuse white surface at a distance of 1.5 cm. When called with mode set to CAL_BLACK, the sensor should have nothing in front of it within a distance of about 2 m. If the calibration command was received successfully, the sensors LED will blink for confirmation.

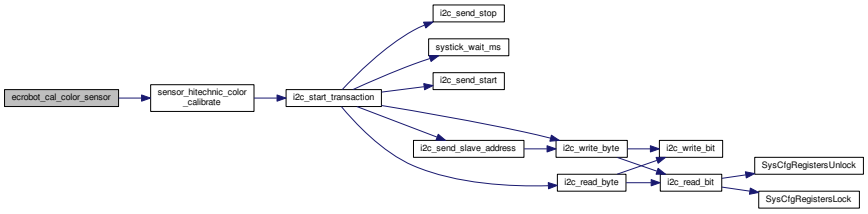
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *mode* | - The mode to calibrate the sensor with: CAL_WHITE (0x43) or CAL_BLACK (0x42) |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:




#### 4.1.2.2 U8 ecrobot_cal_end_compass_sensor ( U8 *port_id* )

Finish the HiTechnic compass sensor calibration.

Read the information in the appropriate start_calibration function first. This function is the third step and finishs the calibration process.

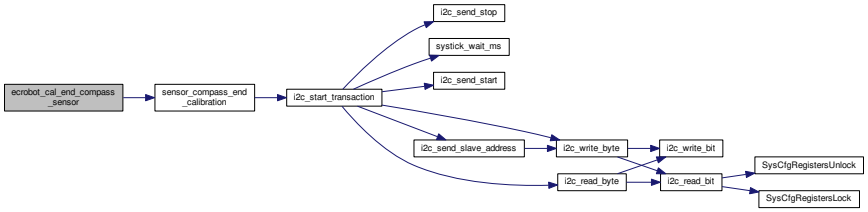
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:




#### 4.1.2.3 void ecrobot_cal_start_compass_sensor ( U8 *port_id* )

Start the HiTechnic compass sensor calibration.

You should calibrate a compass sensor when it is mounted on your robot in the way you want to use it. So the sensor will be calibrated for your specific environment/robot. The calibration adjustment is stored persitent on the sensor itself even if it is turned off. For more information see the HiTechnic documentation. Calibrating the compass sensor takes 3 steps. (1) Call this function. (2) Move the sensor/robot in a circle (540 degrees - 720 degrees within 20 seconds). (3) Call the appropriate end_calibration function.

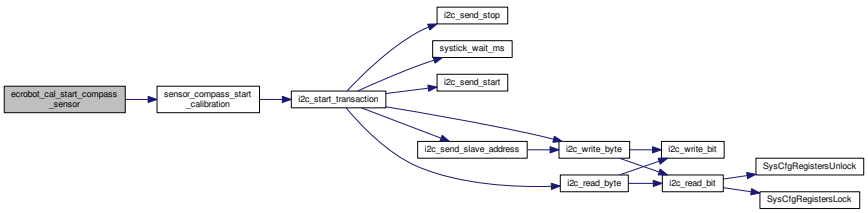
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.4 void ecrobot_get_accel_sensor ( U8 *port_id,* S16 *buf[3]* )

Gets acceleration data in three axes. The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will call sensor_accel_calibrate when called for the first time. The values returned will be relativ to 0.

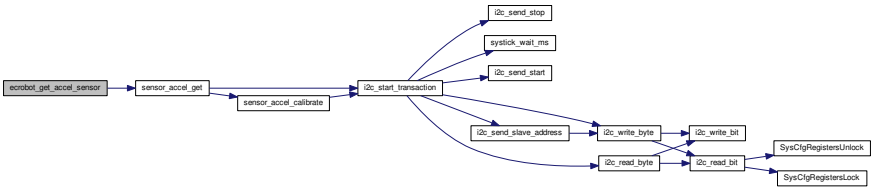
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *buf* | - Buffer to store the meassured values in (the values will be stored in the buffer in the following order: X, Y, Z) |

**Returns**

none

Here is the call graph for this function:



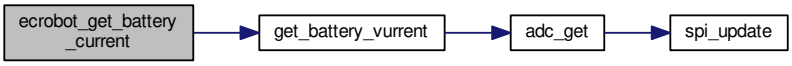

#### 4.1.2.5 U16 ecrobot_get_battery_current ( void )

Get the current of the battery.

**Returns**

The current of the battery

Here is the call graph for this function:



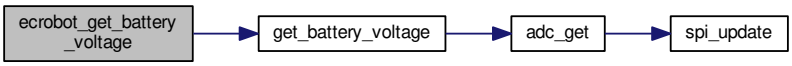

#### 4.1.2.6 U16 ecrobot_get_battery_voltage ( void )

Get the voltage of the battery.

**Returns**

The volatge of the battery

Here is the call graph for this function:



#### 4.1.2.7 void ecrobot_get_color_sensor ( U8 *port_id,* S16 *buf[3]* )

Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.

For best results, the sensor should be calibrated before calling this function.

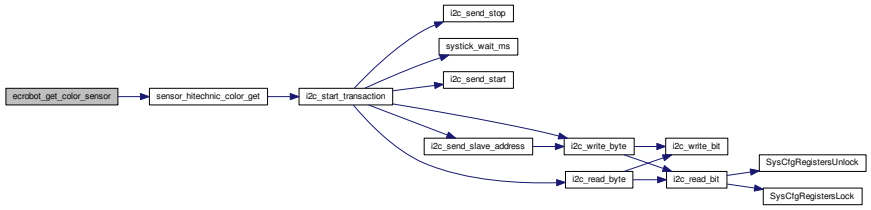
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *buf* | - Buffer to store the received data in (the values will be stored in the following order: red, green, blue) |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.8 S16 ecrobot_get_compass_sensor ( U8 *port_id* )

Read the current direction meassured by the HiTechnic compass sensor at the specified port.

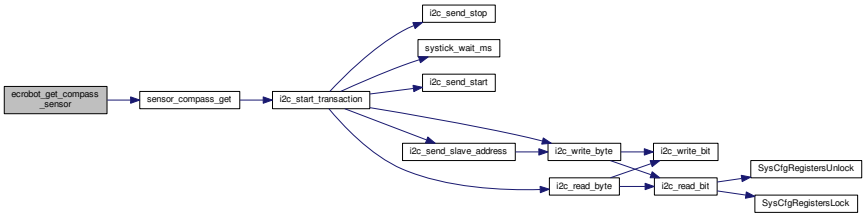
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current direction in degrees meassured by the sensor, ranging from 0 to 360

Here is the call graph for this function:




#### 4.1.2.9 U16 ecrobot_get_gyro_sensor ( U8 *port_id* )

Gets Gyro Sensor raw A/D data. The sensor data has offset value (approximately 600).

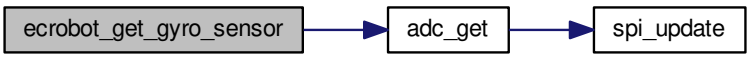
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

raw A/D value

Here is the call graph for this function:




#### 4.1.2.10 S16 ecrobot_get_gyro_sensor_degrees ( U8 *port_id* )

Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.

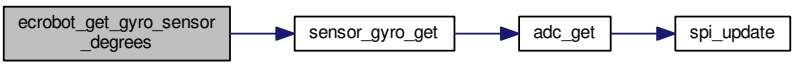
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current rotation meassured by the gyro sensor, ranging from -360 degrees to +360 degrees

Here is the call graph for this function:




#### 4.1.2.11 U16 ecrobot_get_light_sensor ( U8 *port_id* )

Get NXT Light Sensor raw A/D data.

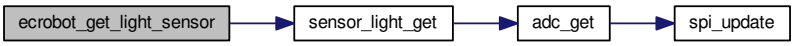
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

A/D raw data(0 to 1023)

Here is the call graph for this function:




#### 4.1.2.12 S32 ecrobot_get_motor_rev ( U8 *port_id* )

Gets Servo Motor revolution value in degree. Wrapper of nxt_motor_get_count.

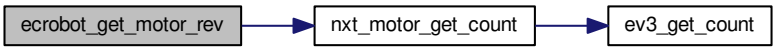
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |

**Returns**

Servo Motors revolution in degree

Here is the call graph for this function:




#### 4.1.2.13 S32 ecrobot_get_sonar_sensor ( U8 *port_id* )

Get Ultrasonic Sensor measurement data in cm.

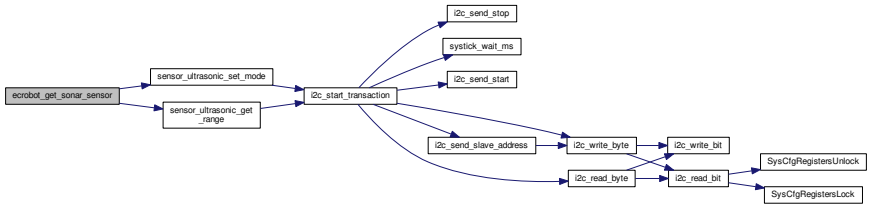
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

distance in cm (0 to 255), -1 (failure)

Here is the call graph for this function:



#### 4.1.2.14 void ecrobot_get_sonar_sensor_single_shot ( U8 *port_id*, U8 *data_buffer[8]* )

Set the mode of the Lego ultrasonic sensor at the specified port to ULTRASONIC_MODE_SINGLE_SHOT. After that get the range of the Lego ultrasonic sensor connected at the specified port and store it in the buffer.

The sensor meassures distances from 0 to 255 in cm. If nothing is located in front of the sensor, the value will be 255. All 8 entries of the array will be values returned by the sensor. If less than 8 objects are detected, some entries will be set to 255.

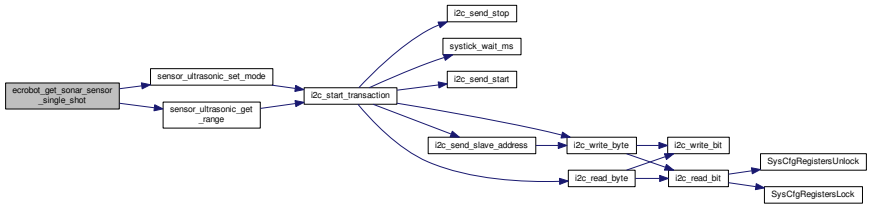
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *data_buffer* | - Buffer to store the result in |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.15 U16 ecrobot_get_sound_sensor ( U8 *port_id* )

Get Sound Sensor raw A/D data.

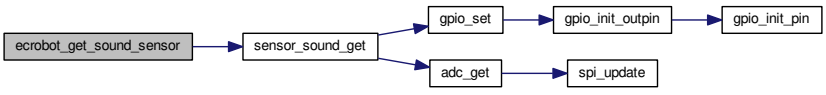
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

A/D raw data(0 to 1023)

Here is the call graph for this function:



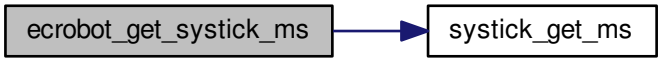

#### 4.1.2.16 U32 ecrobot_get_systick_ms ( void )

Get the current tick in milliseconds.

**Returns**

The current tick in milliseconds

Here is the call graph for this function:




#### 4.1.2.17 float ecrobot_get_temperature_sensor ( U8 *port_id* )

Measures and returns the current temperature value.

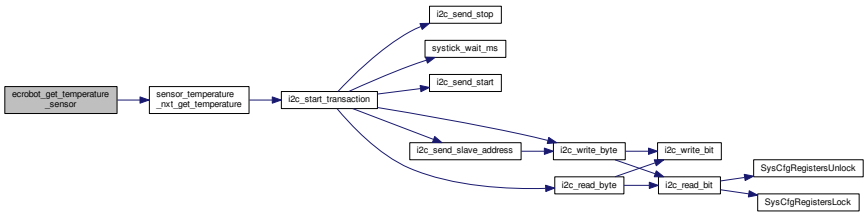
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current temperature given in degrees Celsius

Here is the call graph for this function:




#### 4.1.2.18 U8 ecrobot_get_touch_sensor ( U8 *port_id* )

Get Touch Sensor on/off status.

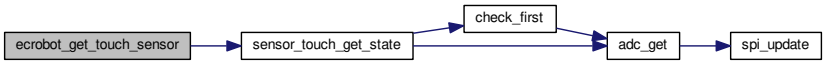
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

1(touched), 0(not touched)

Here is the call graph for this function:




#### 4.1.2.19 U8 ecrobot_getResolution_temperature_sensor ( U8 *port_id* )

Returns the current resolution setting of the sensor.

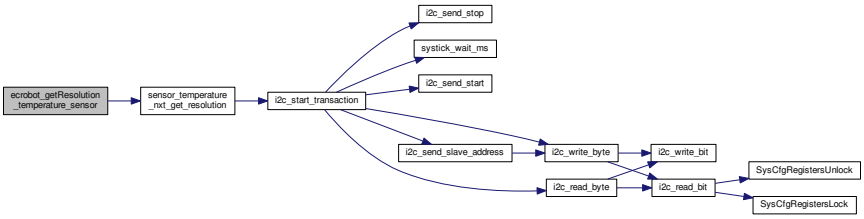
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current resolution. See the macros TEMP9BIT ... TEMP12BIT

Here is the call graph for this function:




#### 4.1.2.20 void ecrobot_init_accel_sensor ( U8 *port_id* )

Initializes a port for I2C communication for Acceleration Sensor. This function should be implemented in the device initialize hook routine.

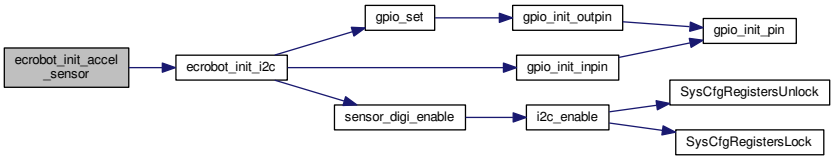
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.21 void ecrobot_init_color_sensor ( U8 *port_id* )

initializes a port for I2C communication for the Color Sensor. This function should be implemented in the device initialize hook routine.

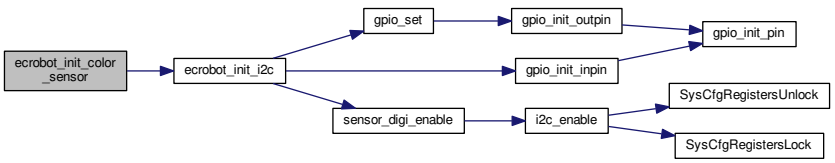
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.22 void ecrobot_init_compass_sensor ( U8 *port_id* )

Initializes a port for I2C communication for Compass Sensor. This function should be implemented in the device initialize hook routine.

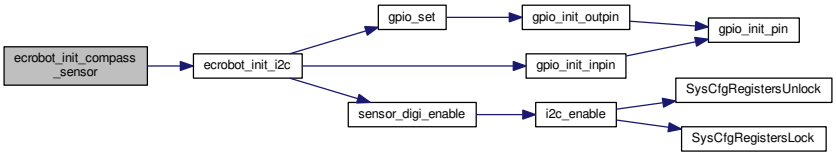
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.23 void ecrobot_init_i2c ( U8 *port_id,* U8 *type* )

Init a NXT sensor port for I2C communication.

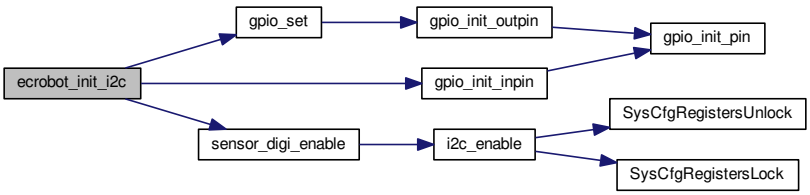
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *type* | - LOWSPEED_9V, LOWSPEED |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.24 void ecrobot_init_sonar_sensor ( U8 *port_id* )

Init a NXT sensor port for Ultrasonic Sensor.

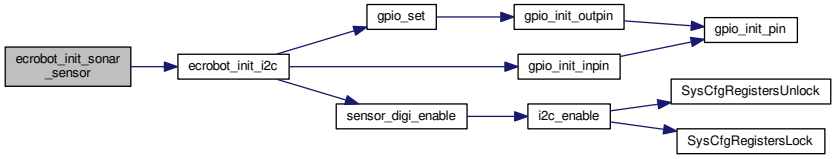
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.25 void ecrobot_init_temperature_sensor ( U8 *port_id* )

Initializes a port for I2C communication for Temperature Sensor.

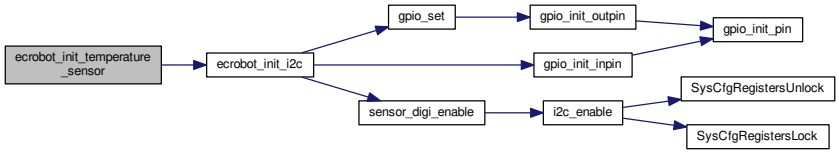
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:



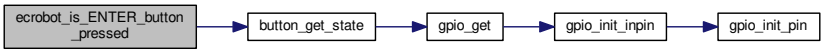

#### 4.1.2.26 U8 ecrobot_is_ENTER_button_pressed ( void )

Returns status of the enter button.

**Returns**

1(is pressed), 0(is not pressed)

Here is the call graph for this function:



#### 4.1.2.27 U8 ecrobot_is_RUN_button_pressed ( void )

Returns status of the run button.

**Returns**

1(is pressed), 0(is not pressed)

Here is the call graph for this function:


#### 4.1.2.28 SINT ecrobot_read_i2c ( U8 *port_id*, U32 *address*, SINT *i2c_reg*, U8 * *buf*, U32 *len* )

Read I2C data.

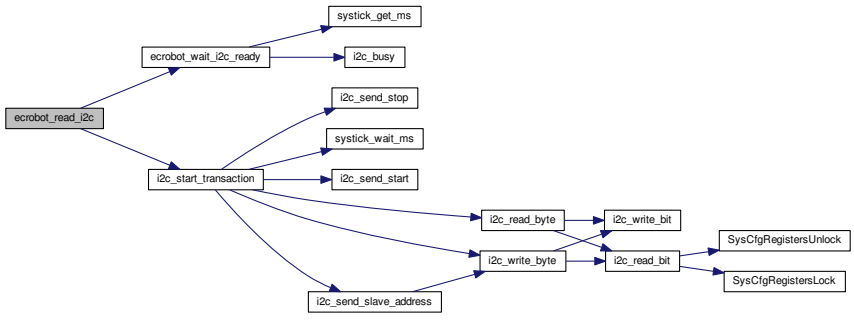
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *address* | - 0x01 to 0x7F (Note that addresses are from 0x01 to 0x7F not even numbers from 0x02 to 0xFE as given in some I2C device specifications. They are 7-bit addresses not 8-bit addresses) |
| *i2c_reg* | - I2C register e.g. 0x42 |
| *buf* | - buffer to return data |
| *len* | - length of the return data |

**Returns**

1(success), 0(failure)

Here is the call graph for this function:




#### 4.1.2.29 SINT ecrobot_send_i2c ( U8 *port_id*, U32 *address*, SINT *i2c_reg*, U8 * *buf*, U32 *len* )

Send I2C data.

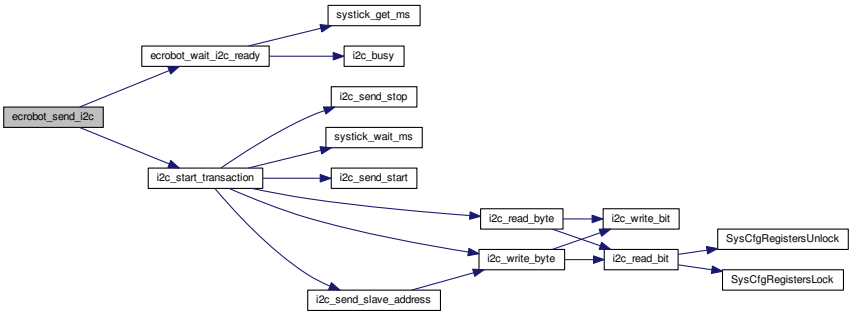
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *address* | - 0x01 to 0x7F (Note that addresses are from 0x01 to 0x7F not even numbers from 0x02 to 0xFE as given in some I2C device specifications. They are 7-bit addresses not 8-bit addresses) |
| *i2c_reg* | - I2C register e.g. 0x42 |
| *buf* | - buffer containing data to send |
| *len* | - length of the data to send |

**Returns**

1(success), 0(failure)

Here is the call graph for this function:




#### 4.1.2.30 void ecrobot_set_light_sensor_active ( U8 *port_id* )

Turn infra-red light on.

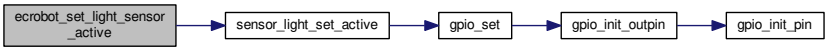
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.31 void ecrobot_set_light_sensor_inactive ( U8 *port_id* )

Turn infra-red light off.

**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:


#### 4.1.2.32 void ecrobot_set_motor_mode_speed ( U8 *port_id,* S32 *mode,* S8 *speed* )

Sets Servo Motor brake mode and PWM value. Wrapper of nxt_motor_set_speed.

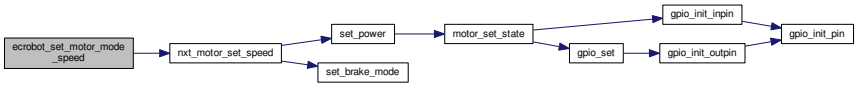
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |
| *mode* | - 0(float), 1(brake) |
| *speed* | - -100 to +100 |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.33 void ecrobot_set_motor_rev ( U8 *port_id,* S32 *rev* )

Sets Servo Motor revolution value in degree. Wrapper of nxt_motor_set_count.

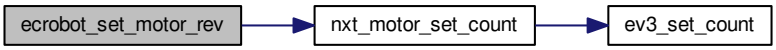
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |
| *rev* | - Servo Motors revolution in degree |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.34 void ecrobot_set_motor_speed ( U8 *port_id,* S8 *speed* )

Sets Servo Motor PWM value. Wrapper of nxt_motor_set_speed, but brake mode is fixed as brake.

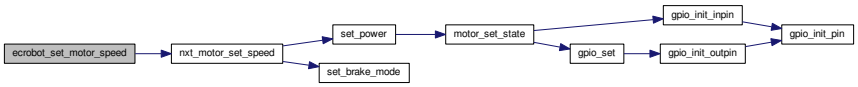
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |
| *speed* | - -100 to +100 |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.35 void ecrobot_setResolution_temperature_sensor ( U8 *port_id,* U8 *resolution* )

Set the resolution for the measured temperature values.

The internal Texas Instruments tmp275 temperature sensor can operate in different resolution modes. Lower resolution is faster. See the TEMP9BIT ... TEMP12BIT macros for more information or the official documentation.

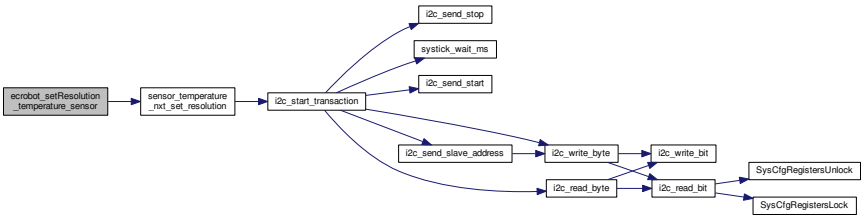
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *resolution* | - The resolution. Use on of the macros TEMP9BIT ... TEMP12BIT |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.36 void ecrobot_term_accel_sensor ( U8 *port_id* )

Terminates I2C communication used for Acceleraton Sensor. This function should be implemented in the device terminate hook routine.

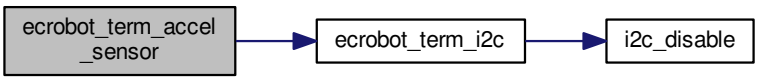
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.37 void ecrobot_term_color_sensor ( U8 *port_id* )

Terminates I2C communication used for the Color Sensor. This function should be implemented in the device terminate hook routine.

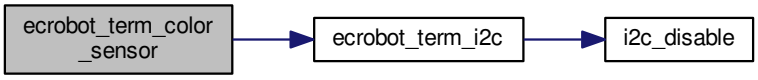
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.38 void ecrobot_term_compass_sensor ( U8 *port_id* )

Terminates I2C communication used for Compass Sensor. This function should be implemented in the device terminate hook routine.

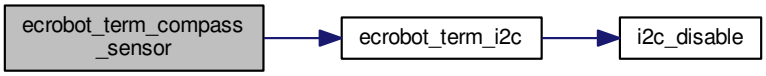
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.39 void ecrobot_term_i2c ( U8 *port_id* )

Terminate a NXT sensor port used for I2C communication.

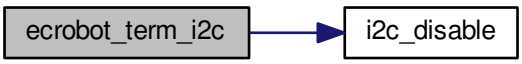
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.40 void ecrobot_term_sonar_sensor ( U8 *port_id* )

Terminate I2C used for for Ultrasonic.

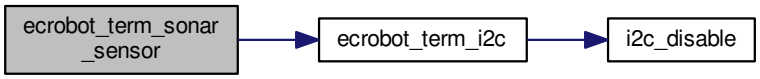
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.41 void ecrobot_term_temperature_sensor ( U8 *port_id* )

Terminates I2C communication used for Temperature Sensor.

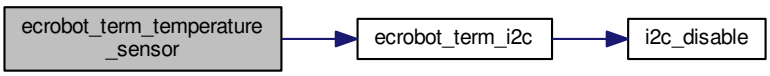
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.1.2.42 U8 ecrobot_wait_i2c_ready ( U8 *port_id*, U32 *wait* )

Wait until I2C communication is ready.

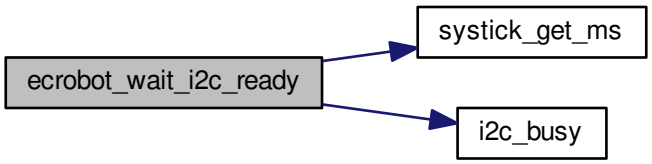
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *wait* | - wait time out in msec |

**Returns**

1(I2C is ready), 0(time out)

Here is the call graph for this function:




#### 4.1.2.43 void ecrobot_wait_ms ( U32 *ms* )

Wait for the specified amount of time.

Waiting with this function is an active waiting which will block until the time has elapsed.

**Parameters**

| | |
|---|---|
| *ms* | - The time to wait in milliseconds |

**Returns**

none

Here is the call graph for this function:


#### 4.1.2.44 void error_function_call_not_implemented ( char * *function_name* )

The internal standard error function for unimplemented ECRobot functions.

**Parameters**

| | |
|---|---|
| *function* | - The function that is currently not implemented |

**Returns**

none

### 4.1.3 Variable Documentation

#### 4.1.3.1 sensor_port_info ports[ ]

A internally used attribute. Needed for port access.

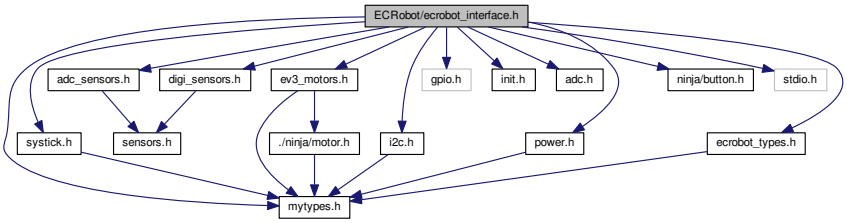
A internally used attribute. Needed for port access.

## 4.2 ECRobot/ecrobot_interface.h File Reference

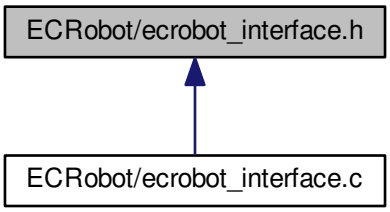
This file contains the function declarations for the users ECRobot-API. This API is used to interact with sensors and actuators.

```
#include "mytypes.h"
#include "systick.h"
#include "adc_sensors.h"
#include "digi_sensors.h"
#include "ev3_motors.h"
#include "i2c.h"
#include "gpio.h"
#include "init.h"
#include "adc.h"
#include "power.h"
#include "ninja/button.h"
#include "stdio.h"
#include "ecrobot_types.h"
```

Include dependency graph for ecrobot_interface.h:




This graph shows which files directly or indirectly include this file:




## Macros

- #define LOWSPEED_9V 1

  *I2C sensor mode used for 9V sensors.*
- #define LOWSPEED 2

  *I2C sensor mode used for sensors with standard voltage.*

## Functions

- S32 ecrobot_get_motor_rev (U8 port_id)

  *Gets Servo Motor revolution value in degree. Wrapper of nxt_motor_get_count.*
- void ecrobot_set_motor_speed (U8 port_id, S8 speed)

  *Sets Servo Motor PWM value. Wrapper of nxt_motor_set_speed, but brake mode is fixed as brake.*
- void ecrobot_set_motor_mode_speed (U8 port_id, S32 mode, S8 speed)

  *Sets Servo Motor brake mode and PWM value. Wrapper of nxt_motor_set_speed.*
- void ecrobot_set_motor_rev (U8 port_id, S32 rev)

  *Sets Servo Motor revolution value in degree. Wrapper of nxt_motor_set_count.*
- U16 ecrobot_get_light_sensor (U8 port_id)

  *Get NXT Light Sensor raw A/D data.*
- void ecrobot_set_light_sensor_active (U8 port_id)

  *Turn infra-red light on.*
- void ecrobot_set_light_sensor_inactive (U8 port_id)

  *Turn infra-red light off.*

- U8 ecrobot_get_touch_sensor (U8 port_id)

  *Get Touch Sensor on/off status.*
- U16 ecrobot_get_sound_sensor (U8 port_id)

  *Get Sound Sensor raw A/D data.*
- void ecrobot_init_i2c (U8 port_id, U8 type)

  *Init a NXT sensor port for I2C communication.*
- U8 ecrobot_wait_i2c_ready (U8 port_id, U32 wait)

  *Wait until I2C communication is ready.*
- SINT ecrobot_send_i2c (U8 port_id, U32 address, SINT i2c_reg, U8 *buf, U32 len)

  *Send I2C data.*
- SINT ecrobot_read_i2c (U8 port_id, U32 address, SINT i2c_reg, U8 *buf, U32 len)

  *Read I2C data.*
- void ecrobot_term_i2c (U8 port_id)

  *Terminate a NXT sensor port used for I2C communication.*
- void ecrobot_init_sonar_sensor (U8 port_id)

  *Init a NXT sensor port for Ultrasonic Sensor.*
- S32 ecrobot_get_sonar_sensor (U8 port_id)

  *Get Ultrasonic Sensor measurement data in cm.*
- void ecrobot_get_sonar_sensor_single_shot (U8 port_id, U8 data_buffer[8])

  *Set the mode of the Lego ultrasonic sensor at the specified port to ULTRASONIC_MODE_SINGLE_SHOT. After that get the range of the Lego ultrasonic sensor connected at the specified port and store it in the buffer.*
- void ecrobot_term_sonar_sensor (U8 port_id)

  *Terminate I2C used for for Ultrasonic.*
- U16 ecrobot_get_gyro_sensor (U8 port_id)

  *Gets Gyro Sensor raw A/D data. The sensor data has offset value (approximately 600).*
- S16 ecrobot_get_gyro_sensor_degrees (U8 port_id)

  *Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.*
- void ecrobot_init_accel_sensor (U8 port_id)

  *Initializes a port for I2C communication for Acceleration Sensor. This function should be implemented in the device initialize hook routine.*
- void ecrobot_get_accel_sensor (U8 port_id, S16 buf[3])

  *Gets acceleration data in three axes. The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will call sensor_accel_calibrate when called for the first time. The values returned will be relativ to 0.*
- void ecrobot_term_accel_sensor (U8 port_id)

  *Terminates I2C communication used for Acceleraton Sensor. This function should be implemented in the device terminate hook routine.*
- void ecrobot_init_color_sensor (U8 port_id)

  *initializes a port for I2C communication for the Color Sensor. This function should be implemented in the device initialize hook routine.*
- U8 ecrobot_cal_color_sensor (U8 port_id, U8 mode)

  *Calibrate the HiTechnic color sensor at the specified port.*
- void ecrobot_get_color_sensor (U8 port_id, S16 buf[3])

  *Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.*
- void ecrobot_term_color_sensor (U8 port_id)

  *Terminates I2C communication used for the Color Sensor. This function should be implemented in the device terminate hook routine.*
- void ecrobot_init_compass_sensor (U8 port_id)

  *Initializes a port for I2C communication for Compass Sensor. This function should be implemented in the device initialize hook routine.*
- S16 ecrobot_get_compass_sensor (U8 port_id)

  *Read the current direction meassured by the HiTechnic compass sensor at the specified port.*
- void ecrobot_term_compass_sensor (U8 port_id)

*Terminates I2C communication used for Compass Sensor. This function should be implemented in the device terminate hook routine.*

- void ecrobot_cal_start_compass_sensor (U8 port_id)

  *Start the HiTechnic compass sensor calibration.*
- U8 ecrobot_cal_end_compass_sensor (U8 port_id)

  *Finish the HiTechnic compass sensor calibration.*
- void ecrobot_init_temperature_sensor (U8 port_id)

  *Initializes a port for I2C communication for Temperature Sensor.*
- float ecrobot_get_temperature_sensor (U8 port_id)

  *Measures and returns the current temperature value.*
- void ecrobot_setResolution_temperature_sensor (U8 port_id, U8 resolution)

  *Set the resolution for the measured temperature values.*
- U8 ecrobot_getResolution_temperature_sensor (U8 port_id)

  *Returns the current resolution setting of the sensor.*
- void ecrobot_term_temperature_sensor (U8 port_id)

  *Terminates I2C communication used for Temperature Sensor.*
- U16 ecrobot_get_battery_voltage (void)

  *Get the voltage of the battery.*
- U16 ecrobot_get_battery_current (void)

  *Get the current of the battery.*
- U32 ecrobot_get_systick_ms (void)

  *Get the current tick in milliseconds.*
- void ecrobot_wait_ms (U32 ms)

  *Wait for the specified amount of time.*
- U8 ecrobot_is_ENTER_button_pressed (void)

  *Returns status of the enter button.*
- U8 ecrobot_is_RUN_button_pressed (void)

  *Returns status of the run button.*

### 4.2.1 Detailed Description

This file contains the function declarations for the users ECRobot-API. This API is used to interact with sensors and actuators.

**Author**

Christian Soward, Tobias Schießl

### 4.2.2 Function Documentation

#### 4.2.2.1 U8 ecrobot_cal_color_sensor ( U8 *port_id,* U8 *mode* )

Calibrate the HiTechnic color sensor at the specified port.

To calibrate the sensor properly, this function has to be called two times. Once with mode set to CAL_WHITE (0x043) and once with mode set to CAL_BLACK (0x42). Calibration information is stored directly on the sensor and will be persistent, even if the sensor is no longer provided with power. When called with mode set to CAL_WHITE, the sensor should be located in front of a diffuse white surface at a distance of 1.5 cm. When called with mode set to CAL_BLACK, the sensor should have nothing in front of it within a distance of about 2 m. If the calibration command was received successfully, the sensors LED will blink for confirmation.

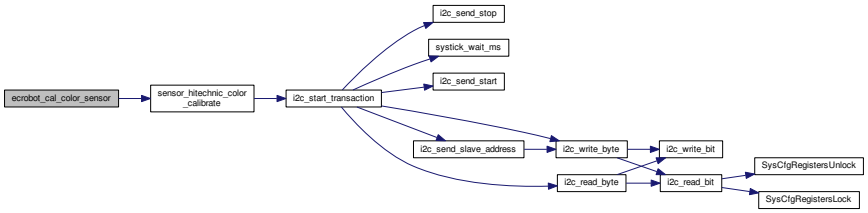
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *mode* | - The mode to calibrate the sensor with: CAL_WHITE (0x43) or CAL_BLACK (0x42) |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:




#### 4.2.2.2 U8 ecrobot_cal_end_compass_sensor ( U8 *port_id* )

Finish the HiTechnic compass sensor calibration.

Read the information in the appropriate start_calibration function first. This function is the third step and finishs the calibration process.

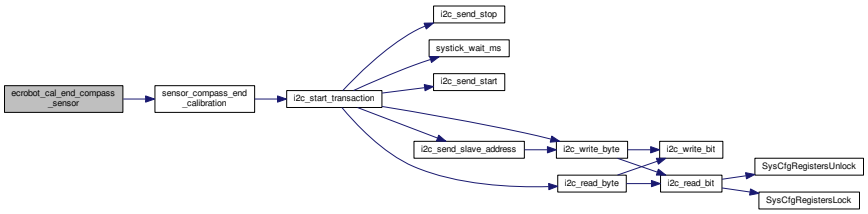
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:




#### 4.2.2.3 void ecrobot_cal_start_compass_sensor ( U8 *port_id* )

Start the HiTechnic compass sensor calibration.

You should calibrate a compass sensor when it is mounted on your robot in the way you want to use it. So the sensor will be calibrated for your specific environment/robot. The calibration adjustment is stored persitent on the sensor itself even if it is turned off. For more information see the HiTechnic documentation. Calibrating the compass sensor takes 3 steps. (1) Call this function. (2) Move the sensor/robot in a circle (540 degrees - 720 degrees within 20 seconds). (3) Call the appropriate end_calibration function.

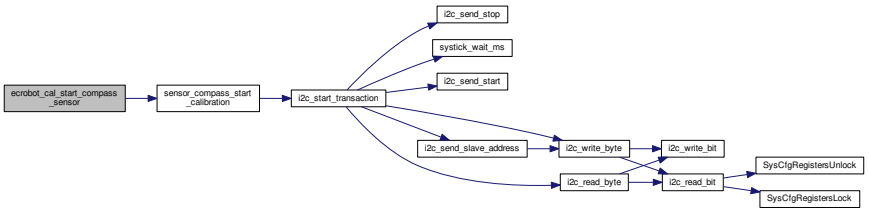
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.4 void ecrobot_get_accel_sensor ( U8 *port_id,* S16 *buf[3]* )

Gets acceleration data in three axes. The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will call sensor_accel_calibrate when called for the first time. The values returned will be relativ to 0.

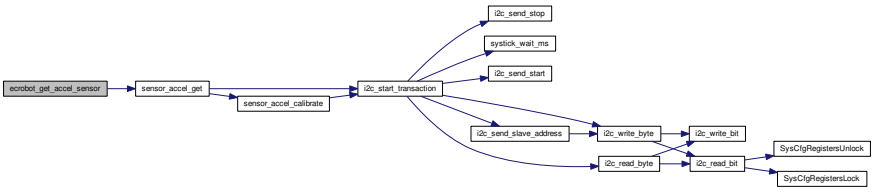
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *buf* | - Buffer to store the meassured values in (the values will be stored in the buffer in the following order: X, Y, Z) |

**Returns**

none

Here is the call graph for this function:



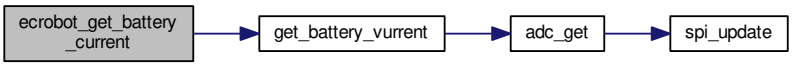

#### 4.2.2.5 U16 ecrobot_get_battery_current ( void )

Get the current of the battery.

**Returns**

The current of the battery

Here is the call graph for this function:



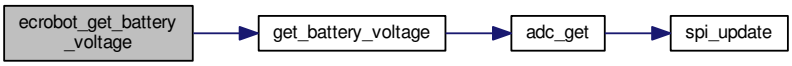

#### 4.2.2.6 U16 ecrobot_get_battery_voltage ( void )

Get the voltage of the battery.

**Returns**

The volatge of the battery

Here is the call graph for this function:




#### 4.2.2.7 void ecrobot_get_color_sensor ( U8 *port_id,* S16 *buf[3]* )

Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.

For best results, the sensor should be calibrated before calling this function.

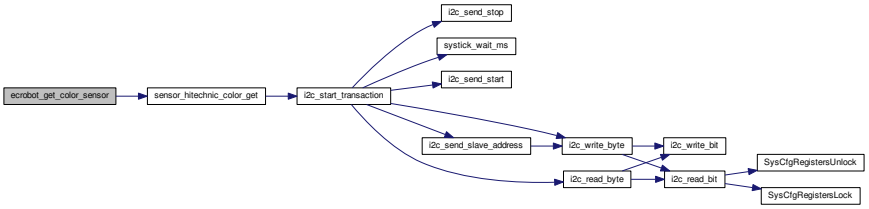
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *buf* | - Buffer to store the received data in (the values will be stored in the following order: red, green, blue) |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.8 S16 ecrobot_get_compass_sensor ( U8 *port_id* )

Read the current direction meassured by the HiTechnic compass sensor at the specified port.

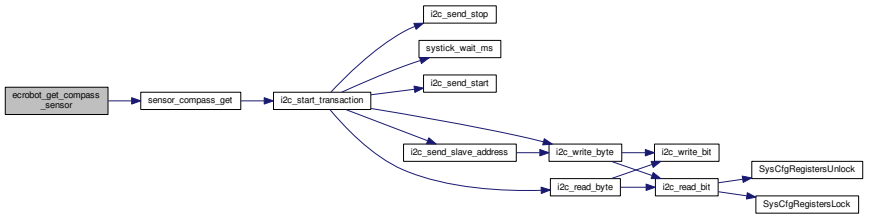
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current direction in degrees meassured by the sensor, ranging from 0 to 360

Here is the call graph for this function:




#### 4.2.2.9 U16 ecrobot_get_gyro_sensor ( U8 *port_id* )

Gets Gyro Sensor raw A/D data. The sensor data has offset value (approximately 600).

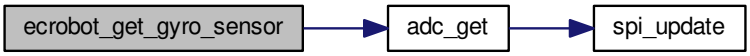
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

raw A/D value

Here is the call graph for this function:




#### 4.2.2.10 S16 ecrobot_get_gyro_sensor_degrees ( U8 *port_id* )

Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.

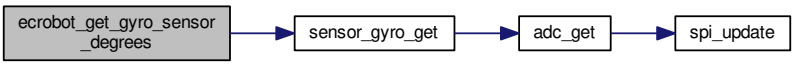
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current rotation meassured by the gyro sensor, ranging from -360 degrees to +360 degrees

Here is the call graph for this function:




#### 4.2.2.11 U16 ecrobot_get_light_sensor ( U8 *port_id* )

Get NXT Light Sensor raw A/D data.

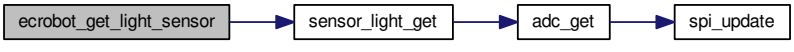
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

A/D raw data(0 to 1023)

Here is the call graph for this function:



#### 4.2.2.12 S32 ecrobot_get_motor_rev ( U8 *port_id* )

Gets Servo Motor revolution value in degree. Wrapper of nxt_motor_get_count.

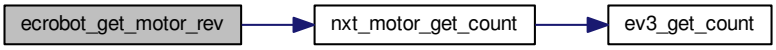
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |

**Returns**

Servo Motors revolution in degree

Here is the call graph for this function:




#### 4.2.2.13 S32 ecrobot_get_sonar_sensor ( U8 *port_id* )

Get Ultrasonic Sensor measurement data in cm.

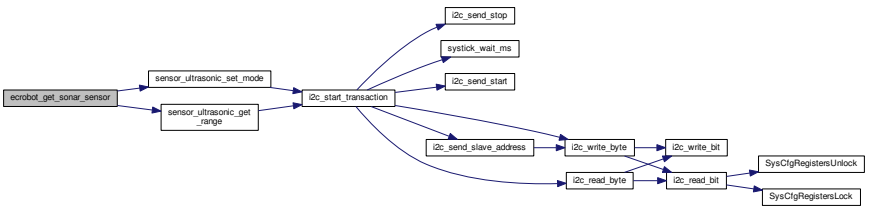
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

distance in cm (0 to 255), -1 (failure)

Here is the call graph for this function:




#### 4.2.2.14 void ecrobot_get_sonar_sensor_single_shot ( U8 *port_id,* U8 *data_buffer[8]* )

Set the mode of the Lego ultrasonic sensor at the specified port to ULTRASONIC_MODE_SINGLE_SHOT. After that get the range of the Lego ultrasonic sensor connected at the specified port and store it in the buffer.

The sensor meassures distances from 0 to 255 in cm. If nothing is located in front of the sensor, the value will be 255. All 8 entries of the array will be values returned by the sensor. If less than 8 objects are detected, some entries will be set to 255.

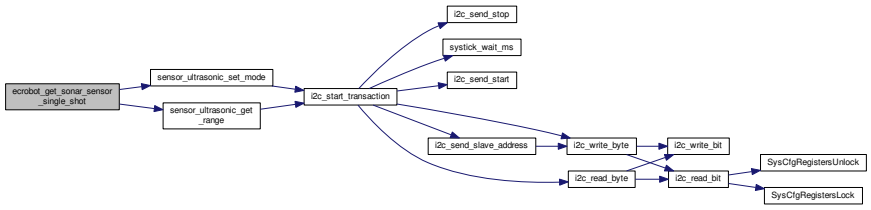
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *data_buffer* | - Buffer to store the result in |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.15 U16 ecrobot_get_sound_sensor ( U8 *port_id* )

Get Sound Sensor raw A/D data.

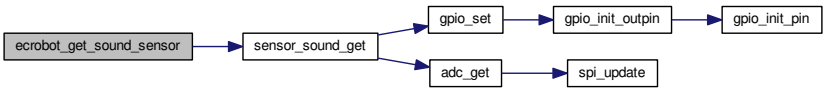
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

A/D raw data(0 to 1023)

Here is the call graph for this function:



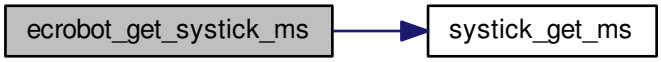

#### 4.2.2.16 U32 ecrobot_get_systick_ms ( void )

Get the current tick in milliseconds.

**Returns**

The current tick in milliseconds

Here is the call graph for this function:




#### 4.2.2.17 float ecrobot_get_temperature_sensor ( U8 *port_id* )

Measures and returns the current temperature value.

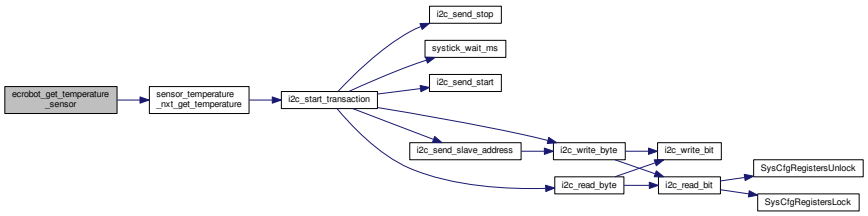
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current temperature given in degrees Celsius

Here is the call graph for this function:




#### 4.2.2.18 U8 ecrobot_get_touch_sensor ( U8 *port_id* )

Get Touch Sensor on/off status.

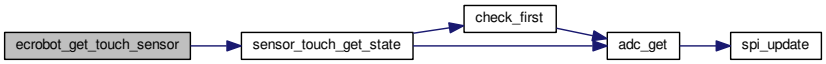
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

1(touched), 0(not touched)

Here is the call graph for this function:




#### 4.2.2.19 U8 ecrobot_getResolution_temperature_sensor ( U8 *port_id* )

Returns the current resolution setting of the sensor.

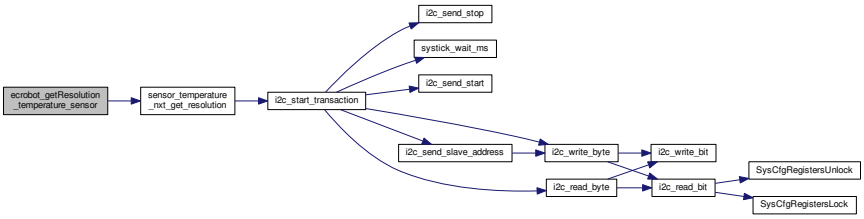
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

The current resolution. See the macros TEMP9BIT ... TEMP12BIT

Here is the call graph for this function:




#### 4.2.2.20 void ecrobot_init_accel_sensor ( U8 *port_id* )

Initializes a port for I2C communication for Acceleration Sensor. This function should be implemented in the device initialize hook routine.

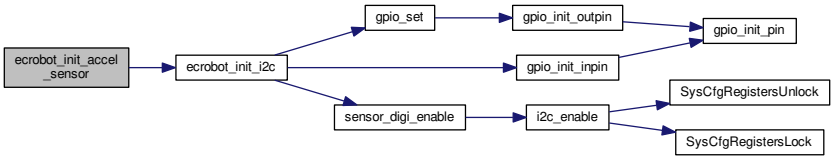
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.21 void ecrobot_init_color_sensor ( U8 *port_id* )

initializes a port for I2C communication for the Color Sensor. This function should be implemented in the device initialize hook routine.

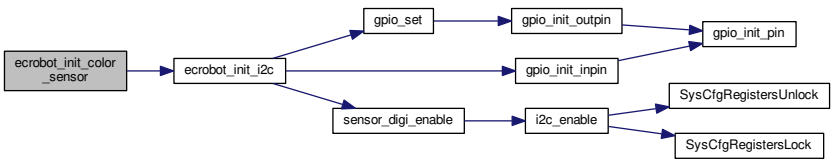
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.22 void ecrobot_init_compass_sensor ( U8 *port_id* )

Initializes a port for I2C communication for Compass Sensor. This function should be implemented in the device initialize hook routine.

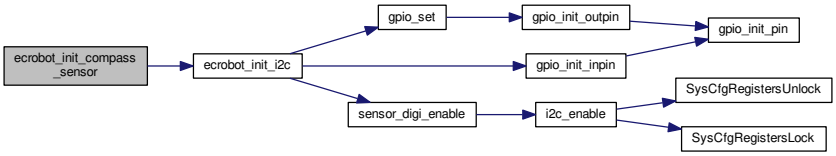
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.23 void ecrobot_init_i2c ( U8 *port_id*, U8 *type* )

Init a NXT sensor port for I2C communication.

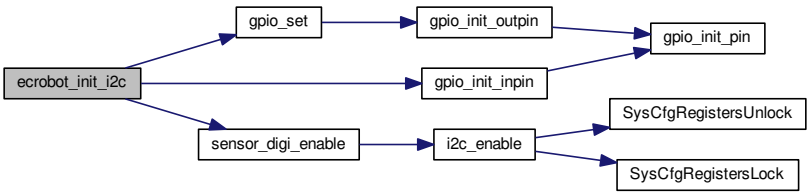
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *type* | - LOWSPEED_9V, LOWSPEED |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.24 void ecrobot_init_sonar_sensor ( U8 *port_id* )

Init a NXT sensor port for Ultrasonic Sensor.

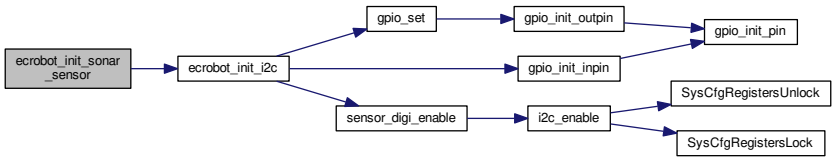
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.25 void ecrobot_init_temperature_sensor ( U8 *port_id* )

Initializes a port for I2C communication for Temperature Sensor.

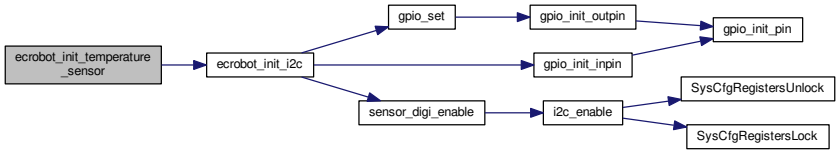
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:



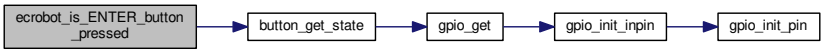

#### 4.2.2.26 U8 ecrobot_is_ENTER_button_pressed ( void )

Returns status of the enter button.

**Returns**

1(is pressed), 0(is not pressed)

Here is the call graph for this function:



#### 4.2.2.27 U8 ecrobot_is_RUN_button_pressed ( void )

Returns status of the run button.

**Returns**

1(is pressed), 0(is not pressed)

Here is the call graph for this function:


#### 4.2.2.28 SINT ecrobot_read_i2c ( U8 *port_id*, U32 *address*, SINT *i2c_reg*, U8 * *buf*, U32 *len* )

Read I2C data.

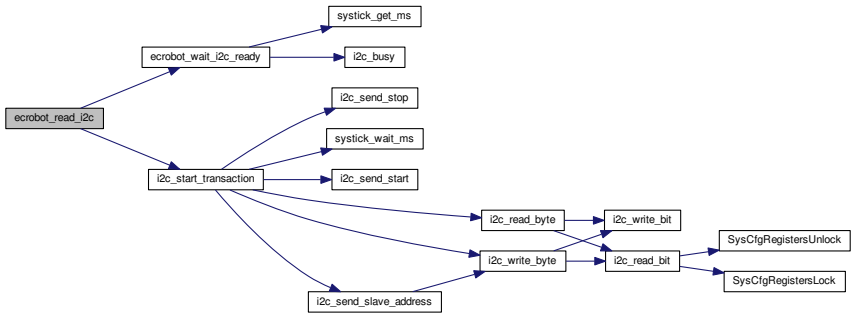
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *address* | - 0x01 to 0x7F (Note that addresses are from 0x01 to 0x7F not even numbers from 0x02 to 0xFE as given in some I2C device specifications. They are 7-bit addresses not 8-bit addresses) |
| *i2c_reg* | - I2C register e.g. 0x42 |
| *buf* | - buffer to return data |
| *len* | - length of the return data |

**Returns**

1(success), 0(failure)

Here is the call graph for this function:




#### 4.2.2.29 SINT ecrobot_send_i2c ( U8 *port_id*, U32 *address*, SINT *i2c_reg*, U8 * *buf*, U32 *len* )

Send I2C data.

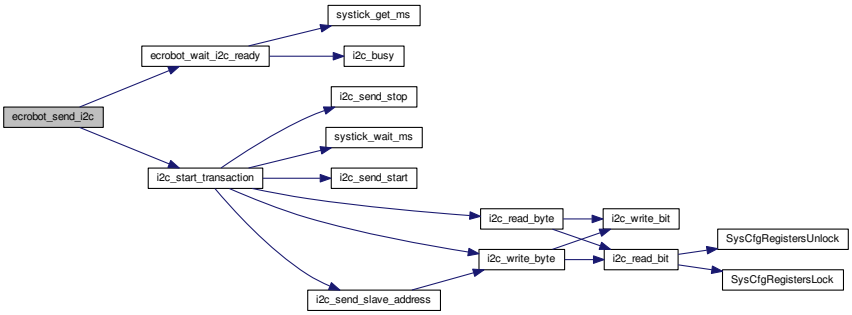
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *address* | - 0x01 to 0x7F (Note that addresses are from 0x01 to 0x7F not even numbers from 0x02 to 0xFE as given in some I2C device specifications. They are 7-bit addresses not 8-bit addresses) |
| *i2c_reg* | - I2C register e.g. 0x42 |
| *buf* | - buffer containing data to send |
| *len* | - length of the data to send |

**Returns**

1(success), 0(failure)

Here is the call graph for this function:




#### 4.2.2.30 void ecrobot_set_light_sensor_active ( U8 *port_id* )

Turn infra-red light on.

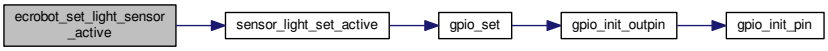
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.31 void ecrobot_set_light_sensor_inactive ( U8 *port_id* )

Turn infra-red light off.

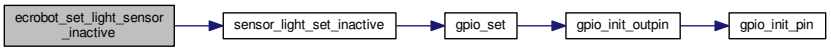
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.32 void ecrobot_set_motor_mode_speed ( U8 *port_id,* S32 *mode,* S8 *speed* )

Sets Servo Motor brake mode and PWM value. Wrapper of nxt_motor_set_speed.

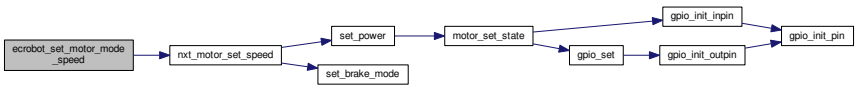
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |
| *mode* | - 0(float), 1(brake) |
| *speed* | - -100 to +100 |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.33 void ecrobot_set_motor_rev ( U8 *port_id,* S32 *rev* )

Sets Servo Motor revolution value in degree. Wrapper of nxt_motor_set_count.

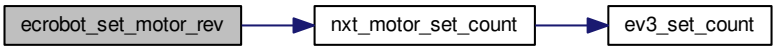
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |
| *rev* | - Servo Motors revolution in degree |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.34 void ecrobot_set_motor_speed ( U8 *port_id,* S8 *speed* )

Sets Servo Motor PWM value. Wrapper of nxt_motor_set_speed, but brake mode is fixed as brake.

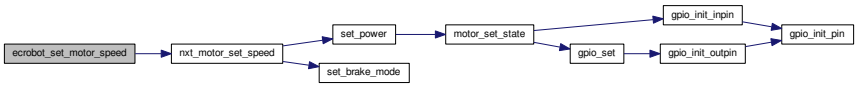
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_1, EV3_PORT_2, EV3_PORT_3, EV3_PORT_4 |
| *speed* | - -100 to +100 |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.35 void ecrobot_setResolution_temperature_sensor ( U8 *port_id,* U8 *resolution* )

Set the resolution for the measured temperature values.

The internal Texas Instruments tmp275 temperature sensor can operate in different resolution modes. Lower resolution is faster. See the TEMP9BIT ... TEMP12BIT macros for more information or the official documentation.

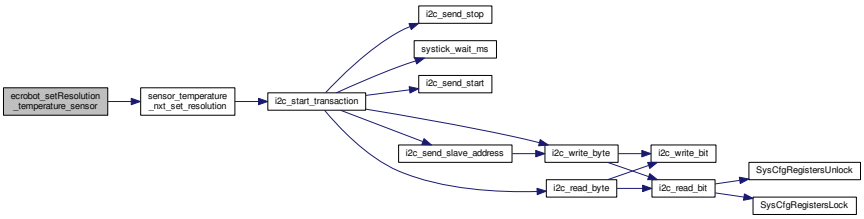
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *resolution* | - The resolution. Use on of the macros TEMP9BIT ... TEMP12BIT |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.36 void ecrobot_term_accel_sensor ( U8 *port_id* )

Terminates I2C communication used for Acceleraton Sensor. This function should be implemented in the device terminate hook routine.

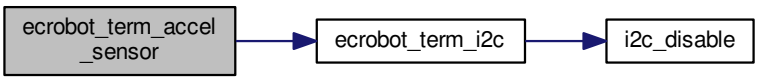
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.37 void ecrobot_term_color_sensor ( U8 *port_id* )

Terminates I2C communication used for the Color Sensor. This function should be implemented in the device terminate hook routine.

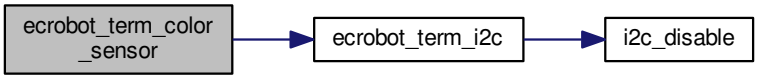
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.38 void ecrobot_term_compass_sensor ( U8 *port_id* )

Terminates I2C communication used for Compass Sensor. This function should be implemented in the device terminate hook routine.

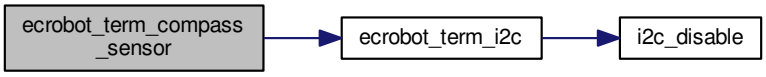
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.39 void ecrobot_term_i2c ( U8 *port_id* )

Terminate a NXT sensor port used for I2C communication.

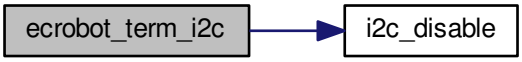
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.40 void ecrobot_term_sonar_sensor ( U8 *port_id* )

Terminate I2C used for for Ultrasonic.

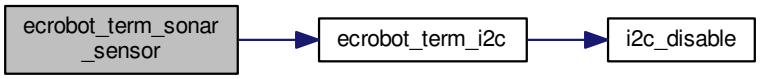
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.41 void ecrobot_term_temperature_sensor ( U8 *port_id* )

Terminates I2C communication used for Temperature Sensor.

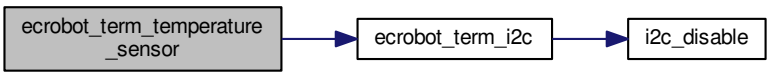
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |

**Returns**

none

Here is the call graph for this function:




#### 4.2.2.42 U8 ecrobot_wait_i2c_ready ( U8 *port_id*, U32 *wait* )

Wait until I2C communication is ready.

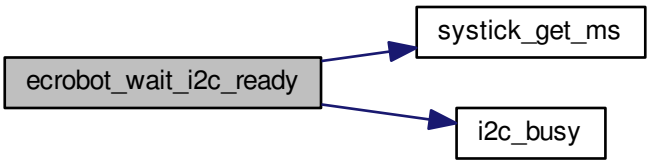
**Parameters**

| | |
|---|---|
| *port_id* | - EV3_PORT_A, EV3_PORT_B, EV3_PORT_C, EV3_PORT_D |
| *wait* | - wait time out in msec |

**Returns**

1(I2C is ready), 0(time out)

Here is the call graph for this function:




#### 4.2.2.43 void ecrobot_wait_ms ( U32 *ms* )

Wait for the specified amount of time.

Waiting with this function is an active waiting which will block until the time has elapsed.

**Parameters**

| | |
|---|---|
| *ms* | - The time to wait in milliseconds |

**Returns**

none

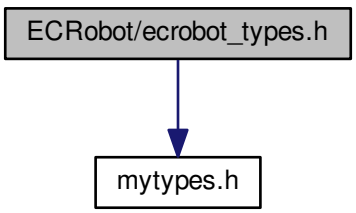
Here is the call graph for this function:


## 4.3 ECRobot/ecrobot_types.h File Reference

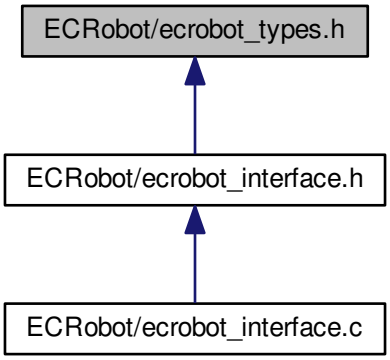
This file contains some typedefs.

```
#include "mytypes.h"
```

Include dependency graph for ecrobot_types.h:



This graph shows which files directly or indirectly include this file:




### Typedefs

- typedef char CHAR

  *The typedef for the type char.*
- typedef unsigned int UINT

  *The typedef for the type unsigned int.*
- typedef signed int SINT

  *The typedef for the type signed int.*
- typedef float F32

  *The typedef for the type float.*
- typedef double F64

  *The typedef for the type double.*

#### 4.3.1 Detailed Description

This file contains some typedefs.

**Author**

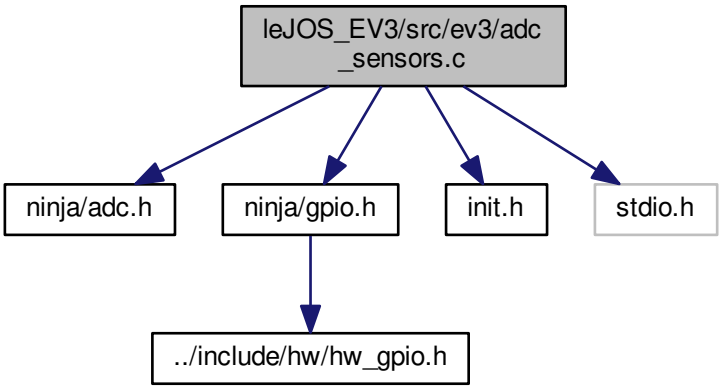
Tobias Schießl

## 4.4 leJOS_EV3/src/ev3/adc_sensors.c File Reference

This file contains the function definitions to read the analogous sensors.

```
#include "ninja/adc.h"
#include "ninja/gpio.h"
#include "init.h"
#include "stdio.h"
```

Include dependency graph for adc_sensors.c:




## Macros

- #define GYRO_SENSOR_OFFSET (2444 >> 2)

  *The 10 bit value the gyro sensor will return if in rest.*

## Functions

- void check_first (int port)

  *Check if we read an analogous sensor for the first time and read the current value if that's the case.*
- int sensor_touch_get_state (int port)

  *Read the state of the NXT touch sensor connected to the specified port.*
- void sensor_light_set_active (int port)

  *Set an NXT light sensor at the specified port active by enabling its LED.*
- void sensor_light_set_inactive (int port)

  *Set an NXT light sensor at the specified port inactive by disabling its LED.*
- unsigned short sensor_light_get (int port)

  *Get the 10 bit value of the NXT light sensor at the specified port.*
- unsigned short sensor_sound_get (int port)

  *Get the 10 bit value of the NXT sound sensor at the specified port.*
- signed short sensor_gyro_get (int port)

  *Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.*

## Variables

- unsigned char first [4] = {1, 1, 1, 1}

  *Array storing the information if an analogous sensor is red for the first time or not.*
- unsigned char sound_sensors_initialized [4] = {0, 0, 0, 0}

  *Array storing the information if a sound sensor is initialized or not.*
- sensor_port_info ports [ ]

  *Array storing information about the 4 sensor ports of the EV3.*

### 4.4.1 Detailed Description

This file contains the function definitions to read the analogous sensors.

**Author**

Tobias Schießl

### 4.4.2 Function Documentation

#### 4.4.2.1 void check_first ( int *port* )

Check if we read an analogous sensor for the first time and read the current value if that's the case.

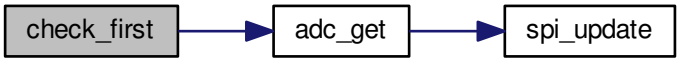
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.4.2.2 signed short sensor_gyro_get ( int *port* )

Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.

The value returned by the sensor will be transformed into the range from -360 degrees to +360 degrees.

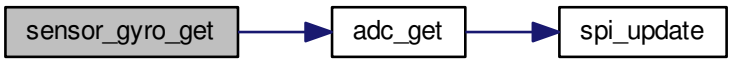
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current rotation meassured by the gyro sensor, ranging from -360 degrees to +360 degrees

Here is the call graph for this function:



#### 4.4.2.3 unsigned short sensor_light_get ( int *port* )

Get the 10 bit value of the NXT light sensor at the specified port.

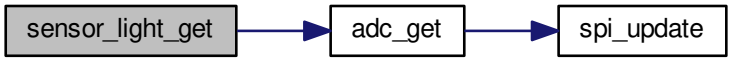
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current value of the light sensor, ranging from 0 (White) to 1023 (Black)

Here is the call graph for this function:




#### 4.4.2.4 void sensor_light_set_active ( int *port* )

Set an NXT light sensor at the specified port active by enabling its LED.

**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:


#### 4.4.2.5 void sensor_light_set_inactive ( int *port* )

Set an NXT light sensor at the specified port inactive by disabling its LED.

**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:


#### 4.4.2.6 unsigned short sensor_sound_get ( int *port* )

Get the 10 bit value of the NXT sound sensor at the specified port.

If this function is called for the first time with the specified port, the sound sensor will be initialized by setting its mode to dB.

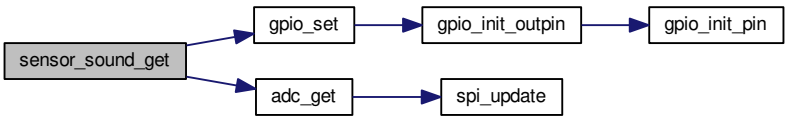
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current value of the sound sensor, ranging from 0 (loud) to 1023 (quiet)

Here is the call graph for this function:




#### 4.4.2.7 int sensor_touch_get_state ( int *port* )

Read the state of the NXT touch sensor connected to the specified port.

If the sensor is red for the first time, the first value is invalid and will be discarded. The current state is red a second time in that case.

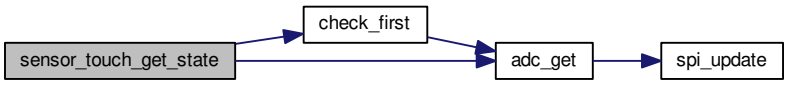
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current state of the touch sensor: 1 if it is pressed, 0 otherwise

Here is the call graph for this function:




### 4.4.3 Variable Documentation

#### 4.4.3.1 sensor_port_info ports[ ]

Array storing information about the 4 sensor ports of the EV3.

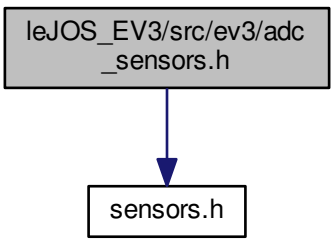
A internally used attribute. Needed for port access.

## 4.5 leJOS_EV3/src/ev3/adc_sensors.h File Reference

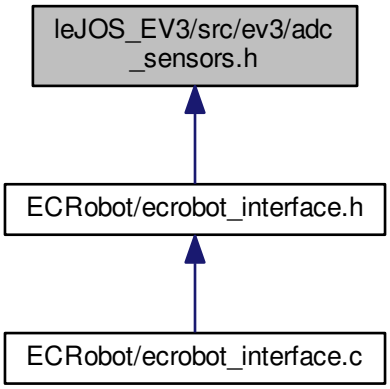
Function declarations to read the analogous sensors.

```
#include "sensors.h"
```

Include dependency graph for adc_sensors.h:



This graph shows which files directly or indirectly include this file:




## Functions

- int sensor_touch_get_state (int port)

  *Read the state of the NXT touch sensor connected to the specified port.*
- void sensor_light_set_active (int port)

  *Set an NXT light sensor at the specified port active by enabling its LED.*
- void sensor_light_set_inactive (int port)

  *Set an NXT light sensor at the specified port inactive by disabling its LED.*
- unsigned short sensor_light_get (int port)

  *Get the 10 bit value of the NXT light sensor at the specified port.*
- unsigned short sensor_sound_get (int port)

  *Get the 10 bit value of the NXT sound sensor at the specified port.*
- signed short sensor_gyro_get (int port)

  *Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.*

### 4.5.1 Detailed Description

Function declarations to read the analogous sensors.

**Author**

Tobias Schießl

### 4.5.2 Function Documentation

#### 4.5.2.1 signed short sensor_gyro_get ( int *port* )

Get the current rotation meassured by the HiTechnic gyro sensor connected at the specified port.

The value returned by the sensor will be transformed into the range from -360 degrees to +360 degrees.

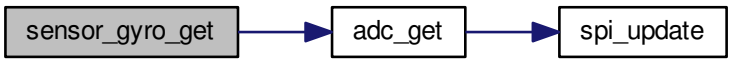
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current rotation meassured by the gyro sensor, ranging from -360 degrees to +360 degrees

Here is the call graph for this function:




#### 4.5.2.2 unsigned short sensor_light_get ( int *port* )

Get the 10 bit value of the NXT light sensor at the specified port.

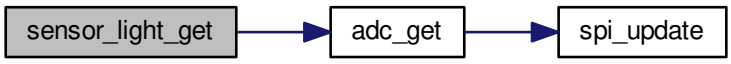
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current value of the light sensor, ranging from 0 (White) to 1023 (Black)

Here is the call graph for this function:




#### 4.5.2.3 void sensor_light_set_active ( int *port* )

Set an NXT light sensor at the specified port active by enabling its LED.

**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:


#### 4.5.2.4 void sensor_light_set_inactive ( int *port* )

Set an NXT light sensor at the specified port inactive by disabling its LED.

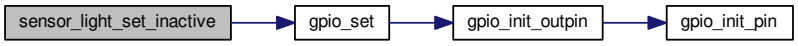
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.5.2.5 unsigned short sensor_sound_get ( int *port* )

Get the 10 bit value of the NXT sound sensor at the specified port.

If this function is called for the first time with the specified port, the sound sensor will be initialized by setting its mode to dB.

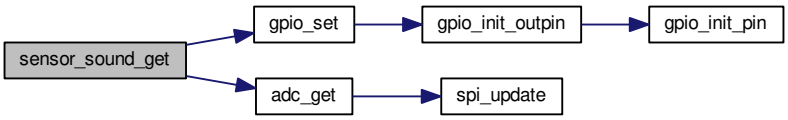
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current value of the sound sensor, ranging from 0 (loud) to 1023 (quiet)

Here is the call graph for this function:




#### 4.5.2.6 int sensor_touch_get_state ( int *port* )

Read the state of the NXT touch sensor connected to the specified port.

If the sensor is red for the first time, the first value is invalid and will be discarded. The current state is red a second time in that case.

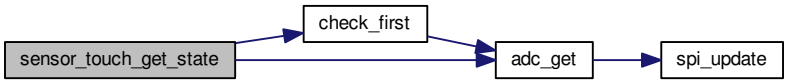
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current state of the touch sensor: 1 if it is pressed, 0 otherwise

Here is the call graph for this function:

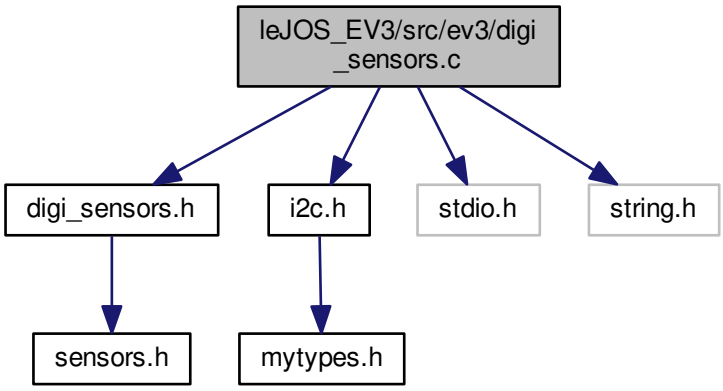



## 4.6 leJOS_EV3/src/ev3/digi_sensors.c File Reference

Function definitions to talk to the digital sensors (I2C sensors)

```
#include "digi_sensors.h"
#include "i2c.h"
#include "stdio.h"
#include "string.h"
```

Include dependency graph for digi_sensors.c:




## Macros

- #define I2C_DEFAULT_SLAVE_ADDRESS 0x01

  *The default slave address used by all I2C sensors so far.*
- #define I2C_DEFAULT_INTERNAL_ADDRESS 0x42

  *The default internal register address which contains the values meassured by the I2C sensors (or the start address in case multiple registers are required)*

## Functions

- void sensor_digi_enable (int port)

  *Enable a digital sensor by configuring the GPIO pins required for I2C communication.*
- void sensor_ultrasonic_get_range (int port, unsigned char data_buffer[8])

  *Get the range of the Lego ultrasonic sensor connected at the specified port.*
- void sensor_ultrasonic_set_mode (int port, unsigned char mode)

  *Set the mode of the Lego ultrasonic sensor at the specified port.*
- unsigned short sensor_compass_get (int port)

  *Read the current direction meassured by the HiTechnic compass sensor at the specified port.*
- void sensor_compass_start_calibration (int port)

  *Start the HiTechnic compass sensor calibration.*
- unsigned short sensor_compass_end_calibration (int port)

  *Stop the HiTechnic compass sensor calibration.*
- void sensor_accel_calibrate (int port)

  *Calibrate the HiTechnic acceleration sensor at the specified port.*
- void sensor_accel_get (int port, short data_buffer[3])

  *Get the 0-relativ acceleration meassured by the HiTechnic acceleration sensor at the specified port.*
- void sensor_accel_get_raw (int port, short data_buffer[3])

  *Get the raw acceleration meassured by the HiTechnic acceleration sensor at the specified port.*
- unsigned char sensor_hitechnic_color_calibrate (int port, int mode)

  *Calibrate the HiTechnic color sensor at the specified port.*

- void sensor_hitechnic_color_get (int port, signed short data_buffer[3])

  *Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.*

- unsigned char sensor_hitechnic_color_get_color_id (int port)

  *Get the ID of the color meassured by the HiTechnic color sensor at the specified port.*

- void sensor_hitechnic_color_id_to_string (unsigned char color_id, char *string_buffer)

  *Transform a HiTechnic color sensor ID into a String.*

- void sensor_temperature_nxt_set_resolution (int port, unsigned char resolution)

  *Set the resolution for the measured temperature values.*

- unsigned char sensor_temperature_nxt_get_resolution (int port)

  *Returns the current resolution setting of the sensor.*

- float sensor_temperature_nxt_get_temperature (int port)

  *Measures and returns the current temperature value.*

## Variables

- unsigned char ultrasonic_modes [4] = {2, 2, 2, 2}

  *Array storing information about the current mode of Lego ultrasonic sensors (0x02 = CONTINOUS MODE is the default mode)*

- short accel_calibration [4][4] = {0}

  *Array storing calibration information for the HiTechnic acceleration sensor.*

### 4.6.1 Detailed Description

Function definitions to talk to the digital sensors (I2C sensors)

**Author**

Tobias Schießl, Christian Soward

### 4.6.2 Function Documentation

#### 4.6.2.1 void sensor_accel_calibrate ( int *port* )

Calibrate the HiTechnic acceleration sensor at the specified port.

The sensor has be at rest when this function is called. Calibrating the sensor means to get the values returned while at rest in order to return values realtive to 0 in calls to sensor_accel_get. If this function is not called directly, it will be triggered on the first call to sensor_accel_get.

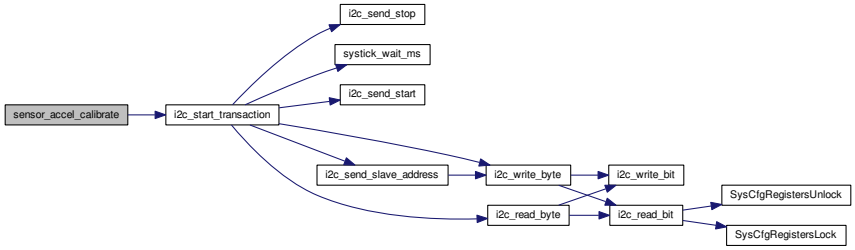
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.2 void sensor_accel_get ( int *port*, short *data_buffer[3]* )

Get the 0-relativ acceleration meassured by the HiTechnic acceleration sensor at the specified port.

The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will call sensor_accel_calibrate when called for the first time. The values returned will be relativ to 0.

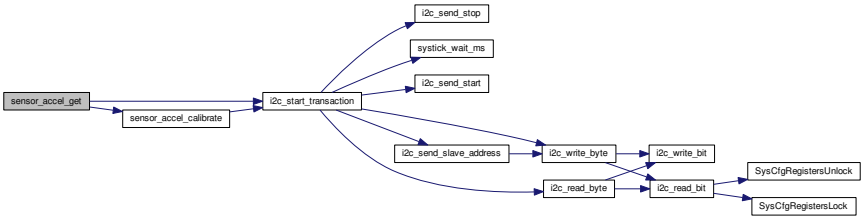
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the meassured values in (the values will be stored in the buffer in the following order: X, Y, Z) |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.3 void sensor_accel_get_raw ( int *port*, short *data_buffer[3]* )

Get the raw acceleration meassured by the HiTechnic acceleration sensor at the specified port.

The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will not call sensor_accel_calibrate and return the raw values (not relativ to 0) returned by the sensor.

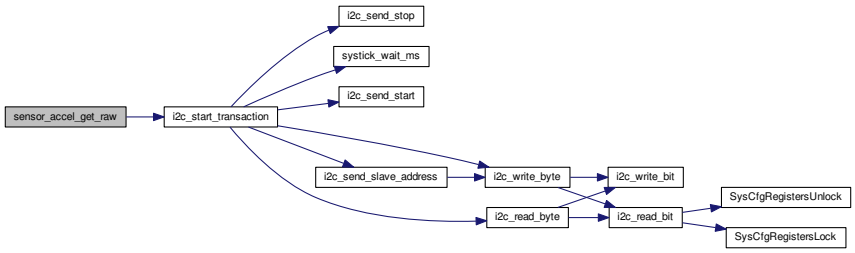
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the meassured values in (the values will be stored in the buffer in the following order: X, Y, Z) |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.4 unsigned short sensor_compass_end_calibration ( int *port* )

Stop the HiTechnic compass sensor calibration.

Read the information in the appropriate start_calibration function first. This function is the third step and finishs the calibration process.

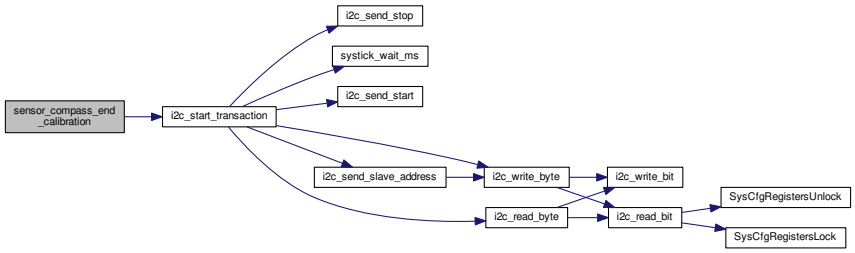
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:




#### 4.6.2.5 unsigned short sensor_compass_get ( int *port* )

Read the current direction meassured by the HiTechnic compass sensor at the specified port.

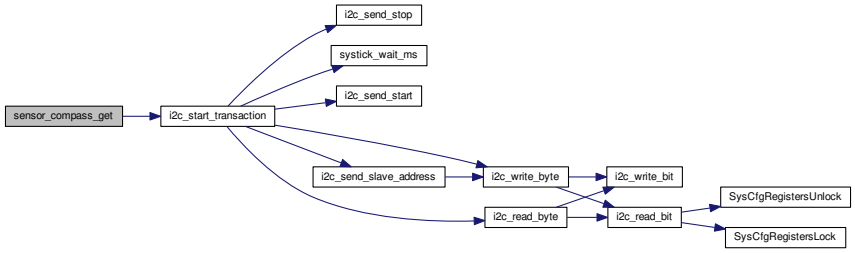
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current direction in degrees meassured by the sensor, ranging from 0 to 360

Here is the call graph for this function:




#### 4.6.2.6 void sensor_compass_start_calibration ( int *port* )

Start the HiTechnic compass sensor calibration.

You should calibrate a compass sensor when it is mounted on your robot in the way you want to use it. So the sensor will be calibrated for your specific environment/robot. The calibration adjustment is stored persitent on the sensor itself. For more information see the HiTechnic documentation. Calibrating the compass sensor takes 3 steps. (1) Call this function. (2) Move the sensor/robot in a circle (540 degrees - 720 degrees within 20 seconds). (3) Call the appropriate end_calibration function.

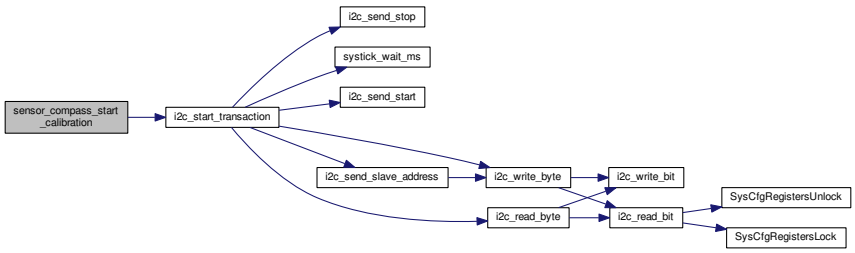
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.7 void sensor_digi_enable ( int *port* )

Enable a digital sensor by configuring the GPIO pins required for I2C communication.

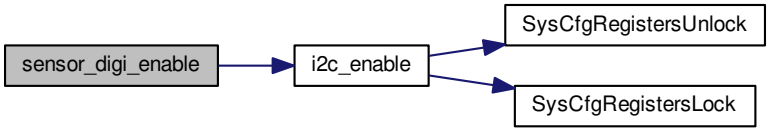
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.8 unsigned char sensor_hitechnic_color_calibrate ( int *port*, int *mode* )

Calibrate the HiTechnic color sensor at the specified port.

To calibrate the sensor properly, this function has to be called two times. Once with mode set to CAL_WHITE (0x043) and once with mode set to CAL_BLACK (0x42). Calibration information is stored directly on the sensor and will be persistent, even if the sensor is no longer provided with power. When called with mode set to CAL_WHITE, the sensor should be located in front of a diffuse white surface at a distance of 1.5 cm. When called with mode set to CAL_BLACK, the sensor should have nothing in front of it within a distance of about 2 m. If the calibration command was received successfully, the sensors LED will blink for confirmation.

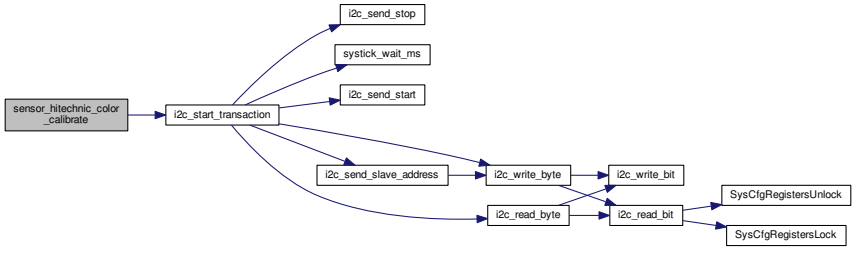
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *mode* | - The mode to calibrate the sensor with: CAL_WHITE (0x43) or CAL_BLACK (0x42) |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:



#### 4.6.2.9 void sensor_hitechnic_color_get ( int *port,* signed short *data_buffer[3]* )

Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.

For best results, the sensor should be calibrated before calling this function.

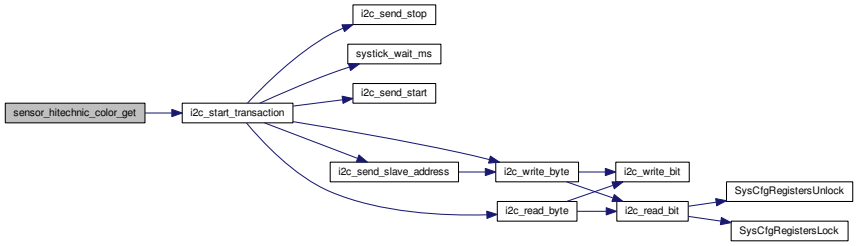
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the received data in (the values will be stored in the following order: red, green, blue) |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.10 unsigned char sensor_hitechnic_color_get_color_id ( int *port* )

Get the ID of the color meassured by the HiTechnic color sensor at the specified port.

For best results, the sensor should be calibrated before calling this function. See digi_sensors.h to see which color is represented by which ID.

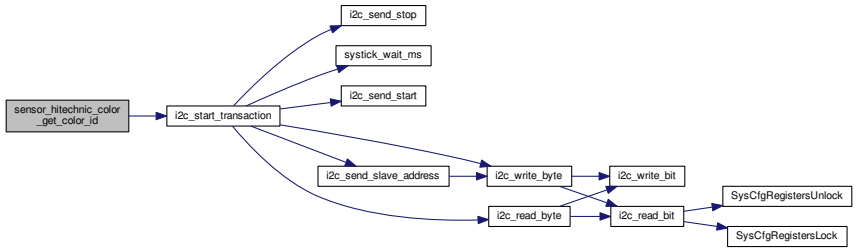
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The ID of the color meassured by the sensor

Here is the call graph for this function:




#### 4.6.2.11 void sensor_hitechnic_color_id_to_string ( unsigned char *color_id,* char * *string_buffer* )

Transform a HiTechnic color sensor ID into a String.

This function was for debug purposes but it might be useful in the future. Therefore it remains part of the code.

**Parameters**

| | |
|---|---|
| *color_id* | - The ID of the color to represent as a String |
| *string_buffer* | - Buffer to store the String representation of the color in |

**Returns**

none

#### 4.6.2.12 unsigned char sensor_temperature_nxt_get_resolution ( int *port* )

Returns the current resolution setting of the sensor.

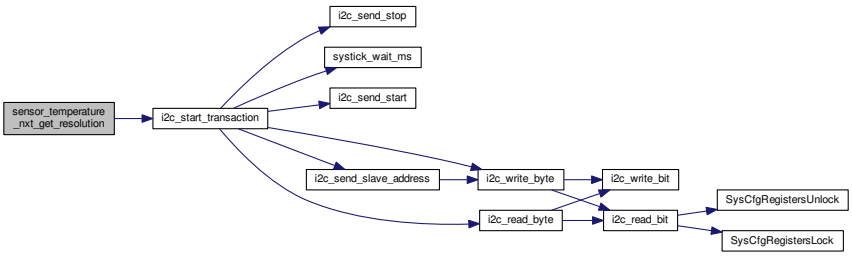
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current resolution. See the macros TEMP9BIT ... TEMP12BIT

Here is the call graph for this function:




#### 4.6.2.13 float sensor_temperature_nxt_get_temperature ( int *port* )

Measures and returns the current temperature value.

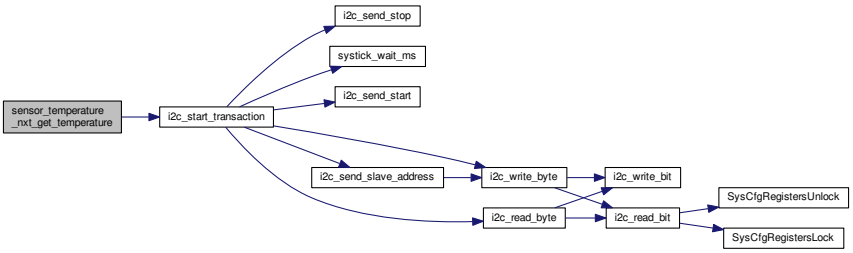
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current temperature given in degrees Celsius

Here is the call graph for this function:




#### 4.6.2.14 void sensor_temperature_nxt_set_resolution ( int *port,* unsigned char *resolution* )

Set the resolution for the measured temperature values.

The internal Texas Instruments tmp275 temperature sensor can operate in different resolution modes. Lower resolution is faster. See the TEMP9BIT ... TEMP12BIT macros for more information or the official documentation.

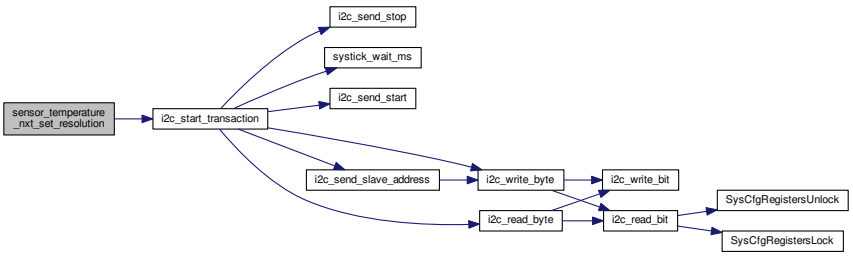
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *resolution* | - The resolution. Use on of the macros TEMP9BIT ... TEMP12BIT |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.15 void sensor_ultrasonic_get_range ( int *port,* unsigned char *data_buffer[8]* )

Get the range of the Lego ultrasonic sensor connected at the specified port.

The sensor meassures distances from 0 to 255 in cm. If nothing is located in front of the sensor, the value will be 255. This function will consider the sensor's current mode. If it is in CONTINOUS MODE (default), only the first value of the array will be valid. The others will be set to 0. If it is in SINGLE SHOT MODE, all 8 entries of the array will be values returned by the sensor. If less than 8 objects are detected, some entries will be set to 255. If it is in OFF MODE, all 8 entries of the array will be set to 0. Note: If the sensor is in SINGLE SHOT MODE, no additional SINGLE SHOT command will be sent in this function - call sensor_ultrasonic_set_mode therefore.

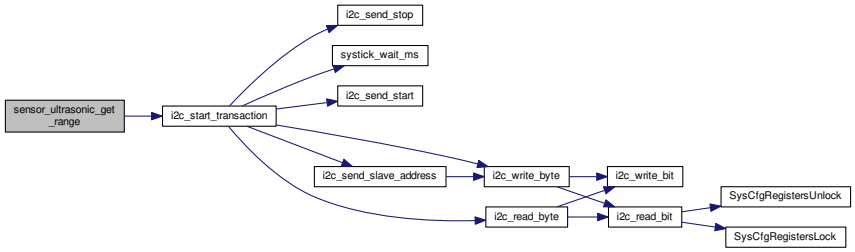
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the result in (depending on the sensor's current mode, not all entries might be filled with values returned by the sensor) |

**Returns**

none

Here is the call graph for this function:




#### 4.6.2.16 void sensor_ultrasonic_set_mode ( int *port*, unsigned char *mode* )

Set the mode of the Lego ultrasonic sensor at the specified port.

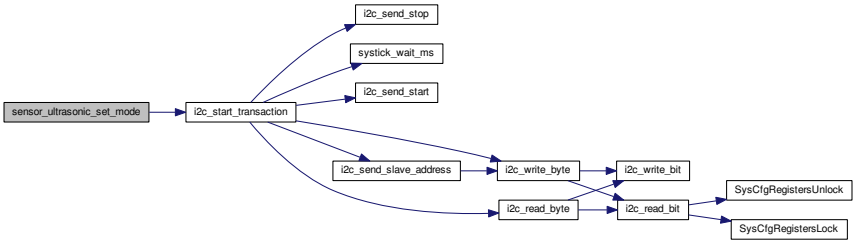
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *mode* | - The mode to set: ULTRASONIC_MODE_OFF (0x00), ULTRASONIC_MODE_SINGLE_↩ SHOT (0x01) or ULTRASONIC_MODE_CONTINOUS (0x02) |

**Returns**

none

Here is the call graph for this function:




### 4.6.3 Variable Documentation

#### 4.6.3.1 short accel_calibration[4][4] = {0}

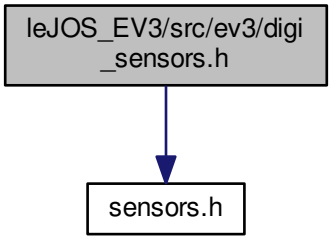
Array storing calibration information for the HiTechnic acceleration sensor.

This information is used to calculate return values for the acceleration sensor which are relativ to 0.

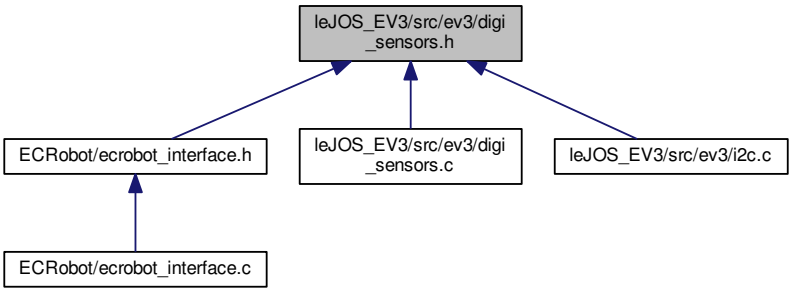
## 4.7 leJOS_EV3/src/ev3/digi_sensors.h File Reference

Function declarations to talk to the digital sensors (I2C sensors)

`#include "sensors.h"`
Include dependency graph for digi_sensors.h:




This graph shows which files directly or indirectly include this file:




### Macros

- #define ULTRASONIC_MODE_CONTINOUS 0x02

  *Ultrasonic sensor mode "CONTINOUS" (sensor will meassure periodically by itself)*
- #define ULTRASONIC_MODE_SINGLE_SHOT 0x01

  *Ultrasonic sensor mode "SINGLE SHOT" (sensor will meassure whenever a new single shot command is received and it will meassure up to 8 objects in front of it)*
- #define ULTRASONIC_MODE_OFF 0x00

  *Ultrasonic sensor mode "OFF" (sensor will not meassure distances)*
- #define CAL_WHITE 0x43

  *Constant value to send to the HiTechnic color sensor in order to start white calibration.*
- #define CAL_BLACK 0x42

*Constant value to send to the HiTechnic color sensor in order to start black calibration.*

- #define HI_TECHNIC_COL_BLACK 0

  *HiTechnic color sensor ID for black.*
- #define HI_TECHNIC_COL_PURPLE 1

  *HiTechnic color sensor ID for black.*
- #define HI_TECHNIC_COL_DARK_BLUE 2

  *HiTechnic color sensor ID for purple.*
- #define HI_TECHNIC_COL_LIGHT_BLUE 3

  *HiTechnic color sensor ID for dark blue.*
- #define HI_TECHNIC_COL_GREEN 4

  *HiTechnic color sensor ID for light blue.*
- #define HI_TECHNIC_COL_YELLOW_GREEN 5

  *HiTechnic color sensor ID for green.*
- #define HI_TECHNIC_COL_YELLOW 6

  *HiTechnic color sensor ID for yellow green.*
- #define HI_TECHNIC_COL_ORANGE 7

  *HiTechnic color sensor ID for yellow.*
- #define HI_TECHNIC_COL_RED 8

  *HiTechnic color sensor ID for orange.*
- #define HI_TECHNIC_COL_RED_PINK 9

  *HiTechnic color sensor ID for red.*
- #define HI_TECHNIC_COL_PINK 10

  *HiTechnic color sensor ID for red pink.*
- #define HI_TECHNIC_COL_GREY1 11

  *HiTechnic color sensor ID for grey.*
- #define HI_TECHNIC_COL_GREY2 12

  *HiTechnic color sensor ID for grey.*
- #define HI_TECHNIC_COL_GREY3 13

  *HiTechnic color sensor ID for grey.*
- #define HI_TECHNIC_COL_GREY4 14

  *HiTechnic color sensor ID for grey.*
- #define HI_TECHNIC_COL_GREY5 15

  *HiTechnic color sensor ID for grey.*
- #define HI_TECHNIC_COL_GREY6 16

  *HiTechnic color sensor ID for grey.*
- #define HI_TECHNIC_COL_WHITE 17

  *HiTechnic color sensor ID for white.*
- #define TEMP9BIT 0x00

  *HiTechnic temperature sensor mode with 9 bit resolution (0.5 degrees C). Average Conversion time within sensor in this mode is 27.5ms.*
- #define TEMP10BIT 0x01

  *HiTechnic temperature sensor mode with 10 bit resolution (0.25 degrees C). Average Conversion time within sensor in this mode is 55ms.*
- #define TEMP11BIT 0x02

  *HiTechnic temperature sensor mode with 11 bit resolution (0.125 degrees C). Average Conversion time within sensor in this mode is 110ms.*
- #define TEMP12BIT 0x03

  *HiTechnic temperature sensor mode with 12 bit resolution (0.0625 degrees C). Average Conversion time within sensor in this mode is 220ms.*

## Functions

- void sensor_digi_enable (int port)

  *Enable a digital sensor by configuring the GPIO pins required for I2C communication.*
- void sensor_ultrasonic_get_range (int port, unsigned char data_buffer[8])

  *Get the range of the Lego ultrasonic sensor connected at the specified port.*
- void sensor_ultrasonic_set_mode (int port, unsigned char mode)

  *Set the mode of the Lego ultrasonic sensor at the specified port.*
- unsigned short sensor_compass_get (int port)

  *Read the current direction meassured by the HiTechnic compass sensor at the specified port.*
- void sensor_compass_start_calibration (int port)

  *Start the HiTechnic compass sensor calibration.*
- unsigned short sensor_compass_end_calibration (int port)

  *Stop the HiTechnic compass sensor calibration.*
- void sensor_accel_calibrate (int port)

  *Calibrate the HiTechnic acceleration sensor at the specified port.*
- void sensor_accel_get (int port, short data_buffer[3])

  *Get the 0-relativ acceleration meassured by the HiTechnic acceleration sensor at the specified port.*
- void sensor_accel_get_raw (int port, short data_buffer[3])

  *Get the raw acceleration meassured by the HiTechnic acceleration sensor at the specified port.*
- unsigned char sensor_hitechnic_color_calibrate (int port, int mode)

  *Calibrate the HiTechnic color sensor at the specified port.*
- void sensor_hitechnic_color_get (int port, signed short data_buffer[3])

  *Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.*
- unsigned char sensor_hitechnic_color_get_color_id (int port)

  *Get the ID of the color meassured by the HiTechnic color sensor at the specified port.*
- void sensor_hitechnic_color_id_to_string (unsigned char color_id, char ∗string_buffer)

  *Transform a HiTechnic color sensor ID into a String.*
- void sensor_temperature_nxt_set_resolution (int port, unsigned char resolution)

  *Set the resolution for the measured temperature values.*
- unsigned char sensor_temperature_nxt_get_resolution (int port)

  *Returns the current resolution setting of the sensor.*
- float sensor_temperature_nxt_get_temperature (int port)

  *Measures and returns the current temperature value.*

### 4.7.1 Detailed Description

Function declarations to talk to the digital sensors (I2C sensors)

**Author**

Tobias Schießl, Christian Soward

### 4.7.2 Function Documentation

#### 4.7.2.1 void sensor_accel_calibrate ( int *port* )

Calibrate the HiTechnic acceleration sensor at the specified port.

The sensor has be at rest when this function is called. Calibrating the sensor means to get the values returned while at rest in order to return values realtive to 0 in calls to sensor_accel_get. If this function is not called directly, it will be triggered on the first call to sensor_accel_get.

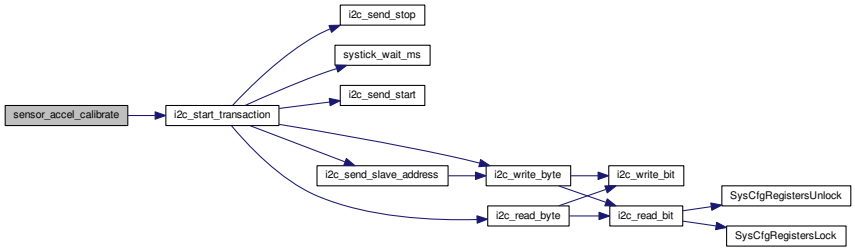
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.2 void sensor_accel_get ( int *port*, short *data_buffer[3]* )

Get the 0-relativ acceleration meassured by the HiTechnic acceleration sensor at the specified port.

The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will call sensor_accel_calibrate when called for the first time. The values returned will be relativ to 0.

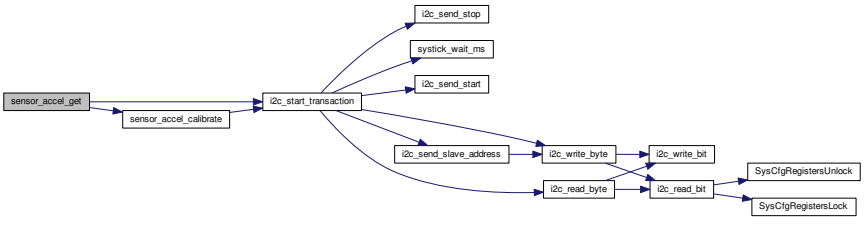
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the meassured values in (the values will be stored in the buffer in the following order: X, Y, Z) |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.3 void sensor_accel_get_raw ( int *port*, short *data_buffer[3]* )

Get the raw acceleration meassured by the HiTechnic acceleration sensor at the specified port.

The sensor meassures the acceleration on all 3 axis (X, Y and Z). This function will not call sensor_accel_calibrate and return the raw values (not relativ to 0) returned by the sensor.

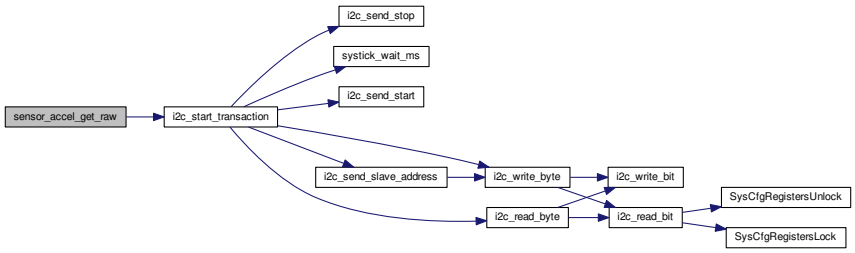
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the meassured values in (the values will be stored in the buffer in the following order: X, Y, Z) |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.4 unsigned short sensor_compass_end_calibration ( int *port* )

Stop the HiTechnic compass sensor calibration.

Read the information in the appropriate start_calibration function first. This function is the third step and finishs the calibration process.

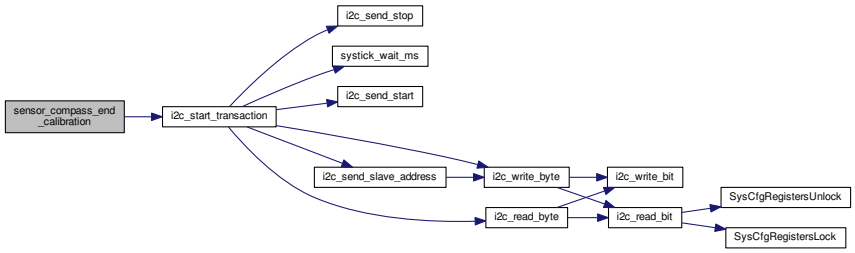
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:




#### 4.7.2.5 unsigned short sensor_compass_get ( int *port* )

Read the current direction meassured by the HiTechnic compass sensor at the specified port.

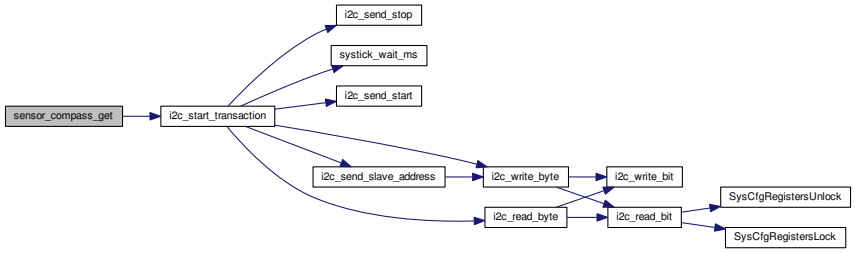
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current direction in degrees meassured by the sensor, ranging from 0 to 360

Here is the call graph for this function:




#### 4.7.2.6 void sensor_compass_start_calibration ( int *port* )

Start the HiTechnic compass sensor calibration.

You should calibrate a compass sensor when it is mounted on your robot in the way you want to use it. So the sensor will be calibrated for your specific environment/robot. The calibration adjustment is stored persitent on the sensor itself. For more information see the HiTechnic documentation. Calibrating the compass sensor takes 3 steps. (1) Call this function. (2) Move the sensor/robot in a circle (540 degrees - 720 degrees within 20 seconds). (3) Call the appropriate end_calibration function.

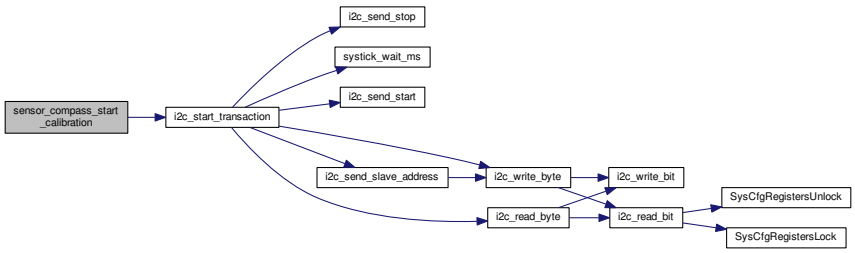
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.7 void sensor_digi_enable ( int *port* )

Enable a digital sensor by configuring the GPIO pins required for I2C communication.

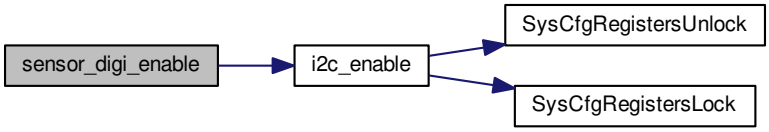
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.8 unsigned char sensor_hitechnic_color_calibrate ( int *port*, int *mode* )

Calibrate the HiTechnic color sensor at the specified port.

To calibrate the sensor properly, this function has to be called two times. Once with mode set to CAL_WHITE (0x043) and once with mode set to CAL_BLACK (0x42). Calibration information is stored directly on the sensor and will be persistent, even if the sensor is no longer provided with power. When called with mode set to CAL_WHITE, the sensor should be located in front of a diffuse white surface at a distance of 1.5 cm. When called with mode set to CAL_BLACK, the sensor should have nothing in front of it within a distance of about 2 m. If the calibration command was received successfully, the sensors LED will blink for confirmation.

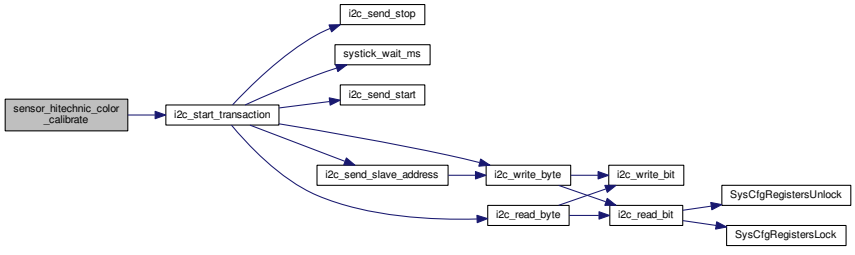
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *mode* | - The mode to calibrate the sensor with: CAL_WHITE (0x43) or CAL_BLACK (0x42) |

**Returns**

1 if the calibration was successful, 0 otherwise

Here is the call graph for this function:



#### 4.7.2.9 void sensor_hitechnic_color_get ( int *port,* signed short *data_buffer[3]* )

Get the red, green and blue values (RGB) meassured by the HiTechnic color sensor at the specified port.

For best results, the sensor should be calibrated before calling this function.

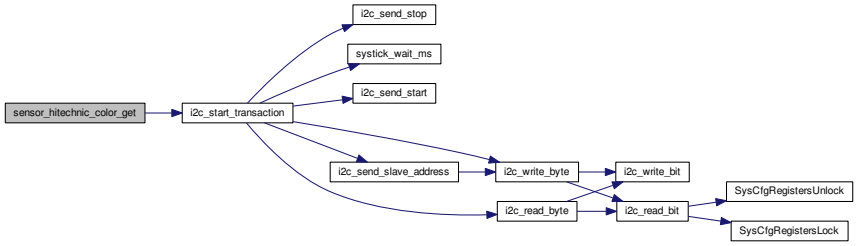
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the received data in (the values will be stored in the following order: red, green, blue) |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.10 unsigned char sensor_hitechnic_color_get_color_id ( int *port* )

Get the ID of the color meassured by the HiTechnic color sensor at the specified port.

For best results, the sensor should be calibrated before calling this function. See digi_sensors.h to see which color is represented by which ID.

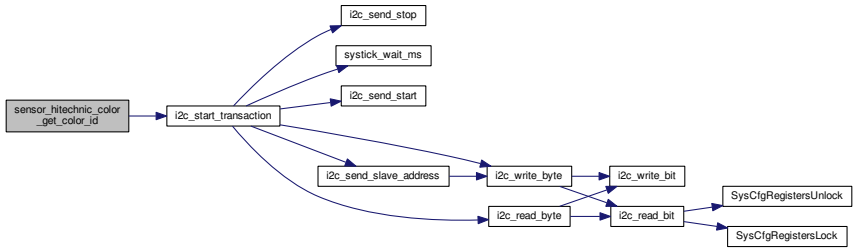
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The ID of the color meassured by the sensor

Here is the call graph for this function:




#### 4.7.2.11 void sensor_hitechnic_color_id_to_string ( unsigned char *color_id,* char * *string_buffer* )

Transform a HiTechnic color sensor ID into a String.

This function was for debug purposes but it might be useful in the future. Therefore it remains part of the code.

**Parameters**

| | |
|---|---|
| *color_id* | - The ID of the color to represent as a String |
| *string_buffer* | - Buffer to store the String representation of the color in |

**Returns**

none

#### 4.7.2.12 unsigned char sensor_temperature_nxt_get_resolution ( int *port* )

Returns the current resolution setting of the sensor.

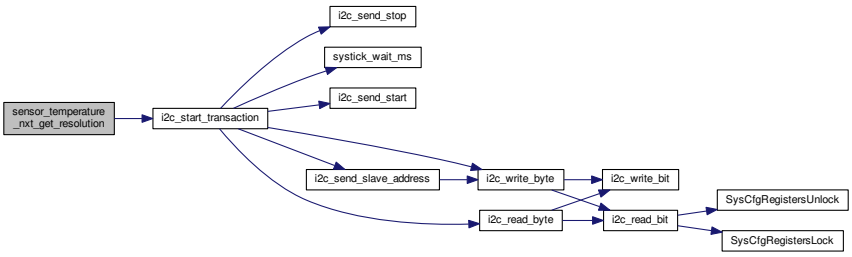
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current resolution. See the macros TEMP9BIT ... TEMP12BIT

Here is the call graph for this function:




#### 4.7.2.13 float sensor_temperature_nxt_get_temperature ( int *port* )

Measures and returns the current temperature value.

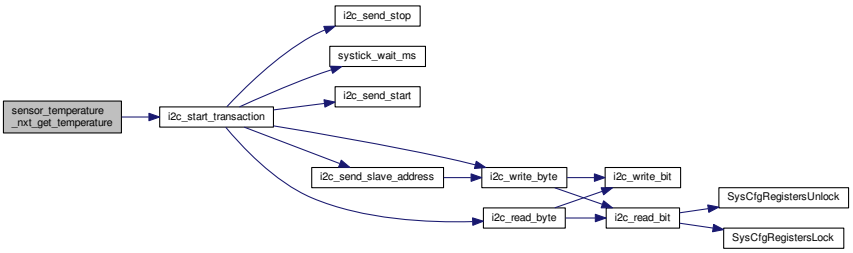
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |

**Returns**

The current temperature given in degrees Celsius

Here is the call graph for this function:




#### 4.7.2.14 void sensor_temperature_nxt_set_resolution ( int *port,* unsigned char *resolution* )

Set the resolution for the measured temperature values.

The internal Texas Instruments tmp275 temperature sensor can operate in different resolution modes. Lower resolution is faster. See the TEMP9BIT ... TEMP12BIT macros for more information or the official documentation.

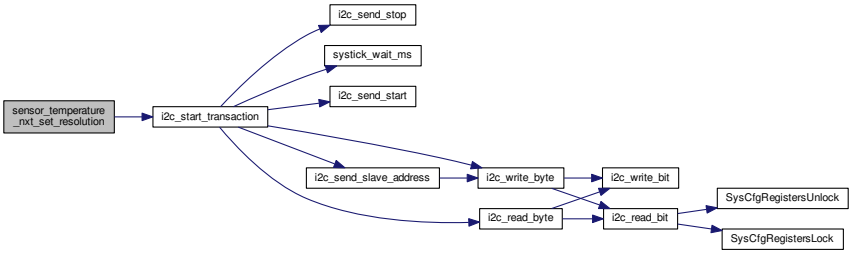
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *resolution* | - The resolution. Use on of the macros TEMP9BIT ... TEMP12BIT |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.15 void sensor_ultrasonic_get_range ( int *port,* unsigned char *data_buffer[8]* )

Get the range of the Lego ultrasonic sensor connected at the specified port.

The sensor meassures distances from 0 to 255 in cm. If nothing is located in front of the sensor, the value will be 255. This function will consider the sensor's current mode. If it is in CONTINOUS MODE (default), only the first value of the array will be valid. The others will be set to 0. If it is in SINGLE SHOT MODE, all 8 entries of the array will be values returned by the sensor. If less than 8 objects are detected, some entries will be set to 255. If it is in OFF MODE, all 8 entries of the array will be set to 0. Note: If the sensor is in SINGLE SHOT MODE, no additional SINGLE SHOT command will be sent in this function - call sensor_ultrasonic_set_mode therefore.

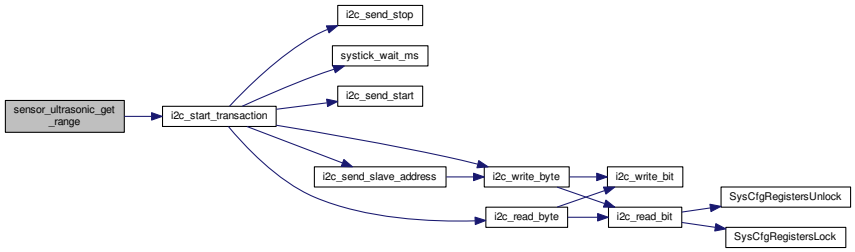
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *data_buffer* | - Buffer to store the result in (depending on the sensor's current mode, not all entries might be filled with values returned by the sensor) |

**Returns**

none

Here is the call graph for this function:




#### 4.7.2.16 void sensor_ultrasonic_set_mode ( int *port*, unsigned char *mode* )

Set the mode of the Lego ultrasonic sensor at the specified port.

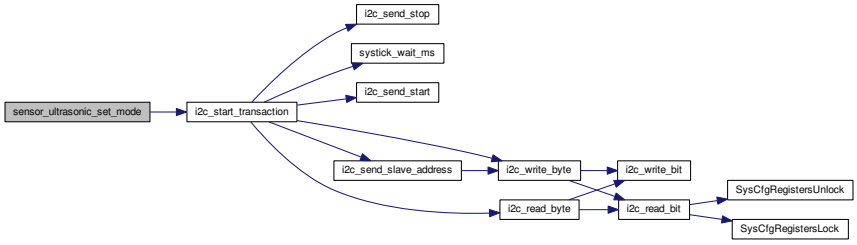
**Parameters**

| | |
|---|---|
| *port* | - The port the sensor is connected to |
| *mode* | - The mode to set: ULTRASONIC_MODE_OFF (0x00), ULTRASONIC_MODE_SINGLE_↩ SHOT (0x01) or ULTRASONIC_MODE_CONTINOUS (0x02) |

**Returns**

none

Here is the call graph for this function:

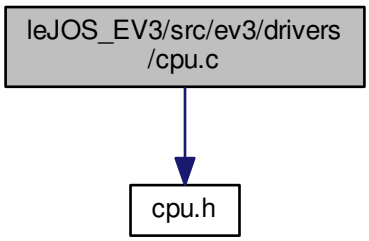



## 4.8 leJOS_EV3/src/ev3/drivers/cpu.c File Reference

This file contains the API definitions for configuring CPU.

```
#include "cpu.h"
```

Include dependency graph for cpu.c:




## Functions

- void CPUSwitchToPrivilegedMode (void)

  *This API can be used to switch from user mode to privileged mode The priviledge mode will be system mode. System mode will share the same resources as user mode, but with privileges.*
- void CPUSwitchToUserMode (void)

  *This API can be used to switch from any previleged mode of ARM to user mode. After this API is called, the program will continue to operate in non-privileged mode, until any exception occurs. After the exception is serviced, execution will continue in user mode.*
- void CPUAbortHandler (void)

  *This API is called when the CPU is aborted or during execution of any undefined instruction. Both IRQ and FIQ will be disabled when the CPU gets an abort and calls this API.*
- unsigned int **CPUIntStatus** (void)
- void **CPUirqd** (void)
- void **CPUirqe** (void)
- void **CPUfiqd** (void)
- void **CPUfiqe** (void)

### 4.8.1 Detailed Description

This file contains the API definitions for configuring CPU.

### 4.8.2 Function Documentation

#### 4.8.2.1 void CPUAbortHandler ( void )

This API is called when the CPU is aborted or during execution of any undefined instruction. Both IRQ and FIQ will be disabled when the CPU gets an abort and calls this API.

**Parameters**

| | |
|---|---|
| *None.* | |

**Returns**

None.

Note : The user can perform error handling such as an immediate reset inside this API if required.

#### 4.8.2.2 void CPUSwitchToPrivilegedMode ( void )

This API can be used to switch from user mode to privileged mode The priviledge mode will be system mode. System mode will share the same resources as user mode, but with privileges.

**Parameters**

| | |
|---|---|
| *None.* | |

**Returns**

None.

Note : All the access to system configuration which needs privileged access can be done after calling this API.

#### 4.8.2.3 void CPUSwitchToUserMode ( void )

This API can be used to switch from any previleged mode of ARM to user mode. After this API is called, the program will continue to operate in non-privileged mode, until any exception occurs. After the exception is serviced, execution will continue in user mode.

**Parameters**

| | |
|---|---|
| *None.* | |

**Returns**

None.

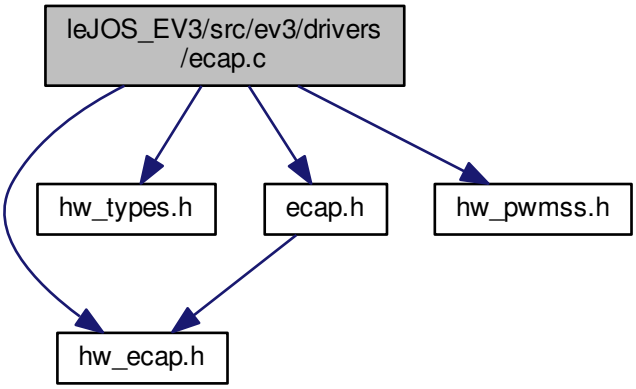
Note : All the access to system configuration which needs privileged access shall be done before this API is called.

## 4.9 leJOS_EV3/src/ev3/drivers/ecap.c File Reference

ECAP APIs.

```
#include "hw_ecap.h"
#include "hw_types.h"
#include "ecap.h"
#include "hw_pwmss.h"
```

Include dependency graph for ecap.c:




## Functions

- void ECAPClockEnable (unsigned int baseAdd)

  *This functions enables clock for ECAP module in PWMSS subsystem.*
- void ECAPClockDisable (unsigned int baseAdd)

  *This functions disables clock for ECAP module in PWMSS subsystem.*
- unsigned int ECAPClockEnableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is enabled or not.*
- unsigned int ECAPClockDisableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is disabled or not.*
- void ECAPCaptureLoadingEnable (unsigned int baseAdd)

  *This function enables capture loading.*
- void ECAPCaptureLoadingDisable (unsigned int baseAdd)

  *This function disables capture loading.*
- void ECAPPrescaleConfig (unsigned int baseAdd, unsigned int prescale)

  *This function configures prescale value.*
- void ECAPOperatingModeSelect (unsigned int baseAdd, unsigned int modeSelect)

  *This function configures ecapture module to operate in capture mode or in APWM mode.*
- unsigned int ECAPTimeStampRead (unsigned int baseAdd, unsigned int capEvtFlag)

  *This function returns time-stamp for a given capture event.*
- void ECAPCounterConfig (unsigned int baseAdd, unsigned int countVal)

  *This function configures the counter register which is used as Capture Time base.*
- void ECAPCapeEvtPolarityConfig (unsigned int baseAdd, unsigned int capEvt1pol, unsigned int capEvt2pol, unsigned int capEvt3pol, unsigned int capEvt4pol)

  *This function configures Capture Event polarity.*
- void ECAPCaptureEvtCntrRstConfig (unsigned int baseAdd, unsigned int CounterRst1, unsigned int CounterRst2, unsigned int CounterRst3, unsigned int CounterRst4)

  *This function enables reset of the counters upon Capture Events.*
- void ECAPContinousModeConfig (unsigned int baseAdd)

  *This function configures ECAP to Continuous mode.*
- void ECAPOneShotModeConfig (unsigned int baseAdd, unsigned int stopVal)

*This function configures ECAP to One-shot mode and also stop value for this mode.*

- void ECAPOneShotREARM (unsigned int baseAdd)

  *This function configures ECAP to One-Short Re-arming.*
- void ECAPAPWMPolarityConfig (unsigned int baseAdd, unsigned int flag)

  *This function configures output polarity for APWM output.*
- void ECAPCounterControl (unsigned int baseAdd, unsigned int flag)

  *This function configures counter to stop or free running based on its input argument flag.*
- void ECAPSyncInOutSelect (unsigned int baseAdd, unsigned int syncIn, unsigned int syncOut)

  *This function configures Sync-In and Sync-Out.*
- void ECAPAPWMCaptureConfig (unsigned int baseAdd, unsigned int compareVal, unsigned int periodVal)

  *When ECAP module is configured in APWM mode capture 1 and capture 2 registers are used as period and compare register.This function configures compare and period values to this register.*
- void ECAPAPWMShadowCaptureConfig (unsigned int baseAdd, unsigned int compareVal, unsigned int periodVal)

  *This function configures the Shadow register.*
- void ECAPCounterPhaseValConfig (unsigned int baseAdd, unsigned int cntPhaseVal)

  *This function configures the counter phase value.*
- void ECAPGlobalIntEnable (unsigned int baseAdd)

  *This function enables the generation of interrupts if any of event interrupt are enable and corresponding event interrupt flag is set.*
- void ECAPIntEnable (unsigned int baseAdd, unsigned int flag)

  *This function enables the specified interrups.*
- void ECAPIntDisable (unsigned int baseAdd, unsigned int flag)

  *This function disables the specified interrups.*
- unsigned int ECAPIntStatus (unsigned int baseAdd, unsigned int flag)

  *This function returns the status specified interrups.*
- void ECAPIntStatusClear (unsigned int baseAdd, unsigned int flag)

  *This function clears of the status specified interrups.*
- unsigned int ECAPPeripheralIdGet (unsigned int baseAdd)

  *This function returns the peripheral ID.*
- void EcapContextSave (unsigned int ecapBase, unsigned int pwmssBase, ECAPCONTEXT *contextPtr)

  *This API can be used to save the register context of ECAP.*
- void EcapContextRestore (unsigned int ecapBase, unsigned int pwmssBase, ECAPCONTEXT *contextPtr)

  *This API can be used to restore the register context of ECAP.*

### 4.9.1 Detailed Description

ECAP APIs.

This file contains the device abstraction layer APIs for ECAP

### 4.9.2 Function Documentation

#### 4.9.2.1 void ECAPAPWMCaptureConfig ( unsigned int *baseAdd,* unsigned int *compareVal,* unsigned int *periodVal* )

When ECAP module is configured in APWM mode capture 1 and capture 2 registers are used as period and compare register.This function configures compare and period values to this register.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *compareVal* | It is the Compare value to be configured. |
| *periodVal* | It is the Period value to be configured. |

**Returns**

None.

#### 4.9.2.2 void ECAPAPWMPolarityConfig ( unsigned int *baseAdd,* unsigned int *flag* )

This function configures output polarity for APWM output.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which determines the output polarity for APWM output.<br>flag can take following macros.<br>ECAP_APWM_ACTIVE_HIGH.<br>ECAP_APWM_ACTIVE_LOW. |

**Returns**

None.

#### 4.9.2.3 void ECAPAPWMShadowCaptureConfig ( unsigned int *baseAdd,* unsigned int *compareVal,* unsigned int *periodVal* )

This function configures the Shadow register.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *compareVal* | It is the Compare value to be configured. |
| *periodVal* | It is the Period value to be configured. |

**Returns**

None.

#### 4.9.2.4 void ECAPCapeEvtPolarityConfig ( unsigned int *baseAdd,* unsigned int *capEvt1pol,* unsigned int *capEvt2pol,* unsigned int *capEvt3pol,* unsigned int *capEvt4pol* )

This function configures Capture Event polarity.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *capEvt1pol* | It determines whether Capture Event1 has to be generated on rising edge or falling edge of pulse. |

| | |
|---|---|
| *capEvt2pol* | It determines whether Capture Event2 has to be generated on rising edge or falling edge of pulse. |
| *capEvt3pol* | It determines whether Capture Event3 has to be generated on rising edge or falling edge of pulse. |
| *capEvt4pol* | It determines whether Capture Event4 has to be generated on rising edge or falling edge of pulse. |

0 - falling edge 1 - rising edge

**Returns**

None.

#### 4.9.2.5 void ECAPCaptureEvtCntrRstConfig ( unsigned int *baseAdd,* unsigned int *CounterRst1,* unsigned int *CounterRst2,* unsigned int *CounterRst3,* unsigned int *CounterRst4* )

This function enables reset of the counters upon Capture Events.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *CounterRst1* | It determines whether counter has to be reset upon Capture Event1. |
| *CounterRst2* | It determines whether counter has to be reset upon Capture Event2. |
| *CounterRst3* | It determines whether counter has to be reset upon Capture Event3. |
| *CounterRst4* | It determines whether counter has to be reset upon Capture Event4. |

0 - Don't reset counter upon capture event.
1 - Reset upon counter capture event.

**Returns**

None.

#### 4.9.2.6 void ECAPCaptureLoadingDisable ( unsigned int *baseAdd* )

This function disables capture loading.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.

#### 4.9.2.7 void ECAPCaptureLoadingEnable ( unsigned int *baseAdd* )

This function enables capture loading.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.

#### 4.9.2.8 void ECAPClockDisable ( unsigned int *baseAdd* )

This functions disables clock for ECAP module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.9.2.9 unsigned int ECAPClockDisableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is disabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is disabled. return's '0' if clocked is not disabled.

#### 4.9.2.10 void ECAPClockEnable ( unsigned int *baseAdd* )

This functions enables clock for ECAP module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.9.2.11 unsigned int ECAPClockEnableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is enabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is enabled. return's '0' if clocked is not enabled.

#### 4.9.2.12 void EcapContextRestore ( unsigned int *ecapBase,* unsigned int *pwmssBase,* ECAPCONTEXT * *contextPtr* )

This API can be used to restore the register context of ECAP.

**Parameters**

| | |
|---|---|
| *ecapBase* | Base address of ECAP instance |
| *pwmssBase* | Base address of the PWM subsystem |

| | |
|---|---|
| *contextPtr* | Pointer to the structure where ECAP register context need to be saved |

**Returns**

None

#### 4.9.2.13 void EcapContextSave ( unsigned int *ecapBase,* unsigned int *pwmssBase,* ECAPCONTEXT * *contextPtr* )

This API can be used to save the register context of ECAP.

**Parameters**

| | |
|---|---|
| *ecapBase* | Base address of ECAP instance |
| *pwmssBase* | Base address of the PWM subsystem |
| *contextPtr* | Pointer to the structure where ECAP register context need to be saved |

**Returns**

None

#### 4.9.2.14 void ECAPContinousModeConfig ( unsigned int *baseAdd* )

This function configures ECAP to Continuous mode.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.
This API is valid only if ECAP is configured to Capture Mode.It has no significance when ECAP is configured in APWM mode.

#### 4.9.2.15 void ECAPCounterConfig ( unsigned int *baseAdd,* unsigned int *countVal* )

This function configures the counter register which is used as Capture Time base.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *countVal* | It is counter value to be configured. |

**Returns**

None.

#### 4.9.2.16 void ECAPCounterControl ( unsigned int *baseAdd,* unsigned int *flag* )

This function configures counter to stop or free running based on its input argument flag.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which determine counter to be configured to stop or free running.<br>flag can take following macros.<br>ECAP_COUNTER_STOP.<br>ECAP_COUNTER_FREE_RUNNING. |

**Returns**

None.

#### 4.9.2.17 void ECAPCounterPhaseValConfig ( unsigned int *baseAdd,* unsigned int *cntPhaseVal* )

This function configures the counter phase value.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *cntPhaseVal* | It is the counter phase value to be programmed for phase lag/lead. |

**Returns**

None.

#### 4.9.2.18 void ECAPGlobalIntEnable ( unsigned int *baseAdd* )

This function enables the generation of interrupts if any of event interrupt are enable and corresponding event interrupt flag is set.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.

#### 4.9.2.19 void ECAPIntDisable ( unsigned int *baseAdd,* unsigned int *flag* )

This function disables the specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the interrupts to be disabled.<br>flag can take following macros. |

ECAP_CEVT1_INT - Enable Capture Event 1 interrupt.
ECAP_CEVT2_INT - Enable Capture Event 2 interrupt.
ECAP_CEVT3_INT - Enable Capture Event 3 interrupt.
ECAP_CEVT4_INT - Enable Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Enable Counter Overflow interrupt.
ECAP_PRDEQ_INT - Enable Period equal interrupt.
ECAP_CMPEQ_INT - Enable Compare equal interrupt.

**Returns**

None.

#### 4.9.2.20 void ECAPIntEnable ( unsigned int *baseAdd,* unsigned int *flag* )

This function enables the specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the interrupts to be enabled.<br>flag can take following macros |

ECAP_CEVT1_INT - Enable Capture Event 1 interrupt.
ECAP_CEVT2_INT - Enable Capture Event 2 interrupt.
ECAP_CEVT3_INT - Enable Capture Event 3 interrupt.
ECAP_CEVT4_INT - Enable Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Enable Counter Overflow interrupt.
ECAP_PRDEQ_INT - Enable Period equal interrupt.
ECAP_CMPEQ_INT - Enable Compare equal interrupt.

**Returns**

None.

#### 4.9.2.21 unsigned int ECAPIntStatus ( unsigned int *baseAdd,* unsigned int *flag* )

This function returns the status specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the status of interrupts to be returned.<br>flag can take following macros. |

ECAP_CEVT1_INT - Status of Capture Event 1 interrupt.
ECAP_CEVT2_INT - Status of Capture Event 2 interrupt.
ECAP_CEVT3_INT - Status of Capture Event 3 interrupt.
ECAP_CEVT4_INT - Status of Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Status of Counter Overflow interrupt.
ECAP_PRDEQ_INT - Status of Period equal interrupt.
ECAP_CMPEQ_INT - Status of Compare equal interrupt.
ECAP_GLOBAL_INT - Global interrupt status.

**Returns**

Status of the specified interrupts.

#### 4.9.2.22 void ECAPIntStatusClear ( unsigned int *baseAdd,* unsigned int *flag* )

This function clears of the status specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the status of interrupts to be cleared.<br>flag can take following macros. |

ECAP_CEVT1_INT - Status of Capture Event 1 interrupt.
ECAP_CEVT2_INT - Status of Capture Event 2 interrupt.
ECAP_CEVT3_INT - Status of Capture Event 3 interrupt.
ECAP_CEVT4_INT - Status of Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Status of Counter Overflow interrupt.
ECAP_PRDEQ_INT - Status of Period equal interrupt.
ECAP_CMPEQ_INT - Status of Compare equal interrupt.

**Returns**

None.

#### 4.9.2.23 void ECAPOneShotModeConfig ( unsigned int *baseAdd,* unsigned int *stopVal* )

This function configures ECAP to One-shot mode and also stop value for this mode.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *stopVal* | It is the number of captures allowed to occur before Capture register(1-4) are frozen.<br>stopVal can take following macros.<br>ECAP_CAPTURE_EVENT1_STOP - stop after Capture Event 1<br>. ECAP_CAPTURE_EVENT2_STOP - stop after Capture Event 2<br>. ECAP_CAPTURE_EVENT3_STOP - stop after Capture Event 3<br>. ECAP_CAPTURE_EVENT4_STOP - stop after Capture Event 4<br>. |

**Returns**

None.
This API is valid only if ECAP is configured to Capture Mode.It has no significance when ECAP is configured in APWM mode.

#### 4.9.2.24 void ECAPOneShotREARM ( unsigned int *baseAdd* )

This function configures ECAP to One-Short Re-arming.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.
When this API is invoked following things happen.

1. Resets Mod4 counter to zero.
2. Un-freezes the Mod4 counter.
3. Enables capture register loads.

#### 4.9.2.25 void ECAPOperatingModeSelect ( unsigned int *baseAdd,* unsigned int *modeSelect* )

This function configures ecapture module to operate in capture mode or in APWM mode.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *modeSelect* | It is the value which determines whether ecapture module to operate in capture mode or in APWM mode.<br>modeSelect can take following macros. |

ECAP_CAPTURE_MODE - Capture Mode.
ECAP_APWM_MODE - APWM Mode.

**Returns**

None.

#### 4.9.2.26 unsigned int ECAPPeripheralIdGet ( unsigned int *baseAdd* )

This function returns the peripheral ID.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

Peripheral ID.

#### 4.9.2.27 void ECAPPrescaleConfig ( unsigned int *baseAdd,* unsigned int *prescale* )

This function configures prescale value.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *prescale* | It is the value which is used to prescale the incoming input. |

prescale can take any integer value between 0 and 62

**Returns**

None.

#### 4.9.2.28 void ECAPSyncInOutSelect ( unsigned int *baseAdd,* unsigned int *syncIn,* unsigned int *syncOut* )

This function configures Sync-In and Sync-Out.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *syncIn* | It is the value which determines whether to disable syncIn or to enable counter to be loaded from CNTPHS register upon a SYNCI signal.<br>syncIn can take following macros.<br>ECAP_SYNC_IN_DISABLE.<br>ECAP_ENABLE_COUNTER - Enables counter to load from CNTPHS register upon SYNCI signal. |
| *syncOut* | It is the value which select type of syncOut signal (i.e select syncIn event to be the Sync-Out signal, select PRD_eq event to be Sync-Out signal).<br><br>`syncOut can take following macros.\n`<br><br>`ECAP_SYNC_IN - Select syncIn event to be the Sync-Out signal.\n`<br>`ECAP_PRD_EQ - Select PRD_eq event to be Sync-Out signal.\n`<br>`ECAP_SYNC_OUT_DISABLE - Disable syncOut signal.\n` |

**Returns**

None.

#### 4.9.2.29 unsigned int ECAPTimeStampRead ( unsigned int *baseAdd,* unsigned int *capEvtFlag* )

This function returns time-stamp for a given capture event.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *capEvtFlag* | It is the value which determines for which capture event time-stam has to returned. |

capEvtFlag can take following macros.

ECAP_CAPTURE_EVENT_1 - Capture Event 1.
ECAP_CAPTURE_EVENT_2 - Capture Event 2.
ECAP_CAPTURE_EVENT_3 - Capture Event 3.
ECAP_CAPTURE_EVENT_4 - Capture Event 4.

**Returns**

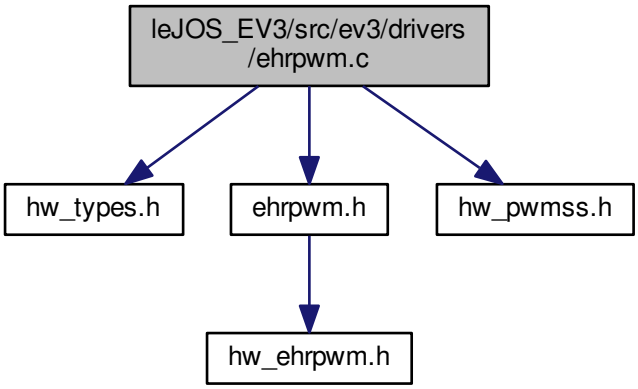
Returns the time-stamp for given capure event.

## 4.10 leJOS_EV3/src/ev3/drivers/ehrpwm.c File Reference

This file contains the device abstraction layer APIs for EHRPWM.

```
#include "hw_types.h"
#include "ehrpwm.h"
#include "hw_pwmss.h"
```

Include dependency graph for ehrpwm.c:




## Functions

- void EHRPWMTimebaseClkConfig (unsigned int baseAddr, unsigned int tbClk, unsigned int moduleClk)

  *This API configures the clock divider of the Time base module. The clock divider can be calculated using the equation TBCLK = SYSCLKOUT/(HSPCLKDIV LKDIV)*
- void EHRPWMPWMOpFreqSet (unsigned int baseAddr, unsigned int tbClk, unsigned int pwmFreq, unsigned int counterDir, bool enableShadowWrite)

  *This API configures the PWM Frequency/Period. The period count determines the period of the final output waveform. For the given period count, in the case of UP and DOWN counter the count value will be loaded as is. In the case of UP_DOWN counter the count is halfed.*
- void EHRPWMTBEmulationModeSet (unsigned int baseAddr, unsigned int mode)

  *This API configures emulation mode. This setting determines the behaviour of Timebase during emulation (debugging).*
- void EHRPWMTimebaseSyncEnable (unsigned int baseAddr, unsigned int tbPhsValue, unsigned int phs↩ CountDir)

  *This API enables the synchronization. When a sync-in event is generated the couter is reloaded with the new value. After sync the counter will use the new value.*
- void EHRPWMTimebaseSyncDisable (unsigned int baseAddr)

  *This API disables the synchronization. Even if sync-in event occurs the count value will not be reloaded.*
- void EHRPWMTriggerSWSync (unsigned int baseAddr)

  *This API generates sw forced sync pulse. This API can be used for testing. When this API is called sync-in will be generated.*
- void EHRPWMSyncOutModeSet (unsigned int baseAddr, unsigned int syncOutMode)

  *This API selects the output sync source. It determines on which of the following event sync-out has to be generated.*
- void EHRPWMWriteTBCount (unsigned int baseAddr, unsigned int tbCount)

  *This API loads the TB counter. The new value is taken immediately.*
- unsigned int EHRPWMReadTBCount (unsigned int baseAddr)

  *This API gets the TB counter current value. The count operation is not affected by the read.*
- unsigned int EHRPWMTBStatusGet (unsigned int baseAddr, unsigned int tbStatusMask)

  *This API gets the TB status as indicated by the tbStatusMask parameter.*
- void EHRPWMTBClearStatus (unsigned int baseAddr, unsigned int tbStatusMask)

  *This API clears the TB status bits indicated by the tbStatusMask parameter.*

- bool EHRPWMLoadCMPA (unsigned int baseAddr, unsigned int CMPAVal, bool enableShadowWrite, unsigned int ShadowToActiveLoadTrigger, bool OverwriteShadowFull)

  *This API loads the CMPA value. When CMPA value equals the counter value, then an event is generated both in the up direction and down direction.*
- bool EHRPWMLoadCMPB (unsigned int baseAddr, unsigned int CMPBVal, bool enableShadowWrite, unsigned int ShadowToActiveLoadTrigger, bool OverwriteShadowFull)

  *This API loads the CMPB value. When CMPB value equals the counter value, then an event is generated both in the up direction and down direction.*
- void EHRPWMConfigureAQActionOnA (unsigned int baseAddr, unsigned int zero, unsigned int period, unsigned int CAUp, unsigned int CADown, unsigned int CBUp, unsigned int CBDown, unsigned int SWForced)

  *This API configures the action to be taken on A by the Action qualifier module upon receiving the events. This will determine the output waveform.*
- void EHRPWMConfigureAQActionOnB (unsigned int baseAddr, unsigned int zero, unsigned int period, unsigned int CAUp, unsigned int CADown, unsigned int CBUp, unsigned int CBDown, unsigned int SWForced)

  *his API configures the action to be taken on B by the Action qualifier module upon receiving the events. This will determine the output waveform.*
- void EHRPWMSWForceA (unsigned int baseAddr)

  *This API triggers the SW forced event on A. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.*
- void EHRPWMSWForceB (unsigned int baseAddr)

  *This API triggers the SW forced event on B. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.*
- void EHRPWMAQContSWForceOnA (unsigned int baseAddr, unsigned int forceVal, unsigned int activeReg↩ ReloadMode)

  *This API forces a value continuously on A. The output can be forced to low or high.*
- void EHRPWMAQContSWForceOnB (unsigned int baseAddr, unsigned int forceVal, unsigned int activeReg↩ ReloadMode)

  *This API forces a value continuously on B. The output can be forced to low or high.*
- void EHRPWMDBSourceSelect (unsigned int baseAddr, unsigned int DBgenSource)

  *This API selects the source for delay blocks in dead band sub-module. The Dead band generator has two sub-modules, one for raising edge delay and the other for falling edge delay. This can be configured when a delay is need between two signals during signal change. The dead band generator is usefull in full-inverters.*
- void EHRPWMDBPolaritySelect (unsigned int baseAddr, unsigned int DBgenPol)

  *This API selects the polarity. This allows to selectively invert one of the delayed signals before it is sent out of the dead-band sub-module.*
- void EHRPWMDBOutput (unsigned int baseAddr, unsigned int DBgenOpMode)

  *This API selects output mode. This allows to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.*
- void EHRPWMDBConfigureRED (unsigned int baseAddr, unsigned int raisingEdgeDelay)

  *This API sets the raising edge delay.*
- void EHRPWMDBConfigureFED (unsigned int baseAddr, unsigned int fallingEdgeDelay)

  *This API sets the Falling edge delay.*
- void EHRPWMConfigureChopperDuty (unsigned int baseAddr, unsigned int dutyCycle)

  *This API configures the chopper duty cyce. In Chopper sub-module the PWM signal is modulated with a carrier signal. Th duty cycle of the carrier signal is configured with this API.*
- void EHRPWMConfigureChopperFreq (unsigned int baseAddr, unsigned int freqDiv)

  *This API configures the chopper frequency. In chopper sub-module the PWM signal is modulated with a carrier signal. The frequency of the carrier signal is configured with this API.*
- void EHRPWMConfigureChopperOSPW (unsigned int baseAddr, unsigned int OSPWCycles)

  *This API configures one shot pulse width. The chopper module is useful in switching operations for pulse transformers. The one-shot block provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on.*
- void EHRPWMChopperEnable (unsigned int baseAddr)

  *This API enables the PWM chopper sub-module.*

- void EHRPWMChopperDisable (unsigned int baseAddr)

  *This API disables the PWM chopper sub-module. This will cause the chopper module to be by-passed.*
- void EHRPWMTZTripEventEnable (unsigned int baseAddr, bool osht_CBC)

  *This API enables the trip event. The trip signals indicates external fault, and the ePWM outputs can be programmed to respond accordingly when faults occur.*
- void EHRPWMTZTripEventDisable (unsigned int baseAddr, bool osht_CBC)

  *This API disable the trip event. The trip events will be ignored.*
- void EHRPWMTZForceAOnTrip (unsigned int baseAddr, unsigned int opValue)

  *This API configures the o/p on A when a trip event is recognized. The output can be set to high or low or high impedence.*
- void EHRPWMTZForceBOnTrip (unsigned int baseAddr, unsigned int opValue)

  *This API configures the o/p on B when a trip event is recognised. The output can be set to high or low or high impedence.*
- void EHRPWMTZIntEnable (unsigned int baseAddr, bool osht_CBC)

  *This API enables the trip interrupt. When trip event occurs the sub-module can be configured to interrupt CPU.*
- void EHRPWMTZIntDisable (unsigned int baseAddr, bool osht_CBC)

  *This API disables the trip interrupt.*
- unsigned int EHRPWMTZFlagGet (unsigned int baseAddr, unsigned int flagToRead)

  *This API returns the flag status requested.*
- void EHRPWMTZFlagClear (unsigned int baseAddr, unsigned int flagToClear)

  *This API clears the flag.*
- void EHRPWMTZSWFrcEvent (unsigned int baseAddr, bool osht_CBC)

  *This API enables to generate SW forced trip.*
- void EHRPWMETIntDisable (unsigned int baseAddr)

  *This API disables the interrupt.*
- void EHRPWMETIntEnable (unsigned int baseAddr)

  *This API enables the interrupt.*
- void EHRPWMETIntSourceSelect (unsigned int baseAddr, unsigned int selectInt)

  *This API selects the interrupt source.*
- void EHRPWMETIntPrescale (unsigned int baseAddr, unsigned int prescale)

  *This API prescales the event on which interrupt is to be generated.*
- bool EHRPWMETEventCount (unsigned int baseAddr)

  *This API returns the number of events occured.*
- bool EHRPWMETIntStatus (unsigned int baseAddr)

  *This API returns the interrupt status.*
- void EHRPWMETIntClear (unsigned int baseAddr)

  *This API clears the interrupt.*
- void EHRPWMETIntSWForce (unsigned int baseAddr)

  *This API forces interrupt to be generated.*
- void EHRPWMLoadTBPHSHR (unsigned int baseAddr, unsigned int TBPHSHRVal)

  *This API loads the HR PHS value.*
- void EHRPWMLoadCMPAHR (unsigned int baseAddr, unsigned int CMPAHRVal, unsigned int ShadowTo↩ ActiveLoadTrigger)

  *This API loads CMPAHR value.*
- void EHRPWMConfigHR (unsigned int baseAddr, unsigned int ctrlMode, unsigned int MEPCtrlEdge)

  *This API configures control mode and edge mode. In also enables the HR sub-module.*
- void EHRPWMHRDisable (unsigned int baseAddr)

  *This API disables the HR sub-module.*
- void EHRPWMClockEnable (unsigned int baseAdd)

  *This functions enables clock for EHRPWM module in PWMSS subsystem.*
- void EHRPWMClockDisable (unsigned int baseAdd)

*This functions enables clock for EHRPWM module in PWMSS subsystem.*

- unsigned int EHRPWMClockEnableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is enabled or not.*

- unsigned int EHRPWMClockDisableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is disabled or not.*

### 4.10.1 Detailed Description

This file contains the device abstraction layer APIs for EHRPWM.

### 4.10.2 Function Documentation

#### 4.10.2.1 void EHRPWMAQContSWForceOnA ( unsigned int *baseAddr,* unsigned int *forceVal,* unsigned int *activeRegReloadMode* )

This API forces a value continuously on A. The output can be forced to low or high.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *forceVal* | Value to be forced |
| *activeReg↩ ReloadMode* | Shadow to active reg load trigger |

**Returns**

None

#### 4.10.2.2 void EHRPWMAQContSWForceOnB ( unsigned int *baseAddr,* unsigned int *forceVal,* unsigned int *activeRegReloadMode* )

This API forces a value continuously on B. The output can be forced to low or high.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *forceVal* | Value to be forced |
| *activeReg↩ ReloadMode* | Shadow to active reg load trigger |

**Returns**

None

#### 4.10.2.3 void EHRPWMChopperDisable ( unsigned int *baseAddr* )

This API disables the PWM chopper sub-module. This will cause the chopper module to be by-passed.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.4 void EHRPWMChopperEnable ( unsigned int *baseAddr* )

This API enables the PWM chopper sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.5 void EHRPWMClockDisable ( unsigned int *baseAdd* )

This functions enables clock for EHRPWM module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.10.2.6 unsigned int EHRPWMClockDisableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is disabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is disabled. return's '0' if clocked is not disabled.

#### 4.10.2.7 void EHRPWMClockEnable ( unsigned int *baseAdd* )

This functions enables clock for EHRPWM module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.10.2.8 unsigned int EHRPWMClockEnableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is enabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is enabled. return's '0' if clocked is not enabled.

#### 4.10.2.9 void EHRPWMConfigHR ( unsigned int *baseAddr,* unsigned int *ctrlMode,* unsigned int *MEPCtrlEdge* )

This API configures control mode and edge mode. In also enables the HR sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *ctrlMode* | phase control or duty control |
| *MEPCtrlEdge* | Edge on which MEP to be applied (raising, falling, both) |

**Returns**

None

#### 4.10.2.10 void EHRPWMConfigureAQActionOnA ( unsigned int *baseAddr,* unsigned int *zero,* unsigned int *period,* unsigned int *CAUp,* unsigned int *CADown,* unsigned int *CBUp,* unsigned int *CBDown,* unsigned int *SWForced* )

This API configures the action to be taken on A by the Action qualifier module upon receiving the events. This will determine the output waveform.

**Parameters**

| | |
|---|---|
| *zero* | Action to be taken when CTR = 0 |
| *period* | Action to be taken when CTR = PRD |
| *CAUp* | Action to be taken when CTR = CAUp |
| *CADown* | Action to be taken when CTR = CADown |
| *CBUp* | Action to be taken when CTR = CBUp |
| *CBDown* | Action to be taken when CTR = CBDown |
| *SWForced* | Action to be taken when SW forced event has been generated<br>Possible values for the actions are<br>– EHRPWM_XXXX_XXXX_DONOTHING \n<br>– EHRPWM_XXXX_XXXX_CLEAR \n<br>– EHRPWM_XXXX_XXXX_SET \n<br>– EHRPWM_XXXX_XXXX_TOGGLE \n |

**Returns**

None

#### 4.10.2.11 void EHRPWMConfigureAQActionOnB ( unsigned int *baseAddr,* unsigned int *zero,* unsigned int *period,* unsigned int *CAUp,* unsigned int *CADown,* unsigned int *CBUp,* unsigned int *CBDown,* unsigned int *SWForced* )

his API configures the action to be taken on B by the Action qualifier module upon receiving the events. This will determine the output waveform.

**Parameters**

| | |
|---|---|
| *zero* | Action to be taken when CTR = 0 |
| *period* | Action to be taken when CTR = PRD |
| *CAUp* | Action to be taken when CTR = CAUp |
| *CADown* | Action to be taken when CTR = CADown |
| *CBUp* | Action to be taken when CTR = CBUp |
| *CBDown* | Action to be taken when CTR = CBDown |
| *SWForced* | Action to be taken when SW forced event has been generated<br>`Possible values for the actions are`<br>`- EHRPWM_XXXX_XXXX_DONOTHING \n`<br>`- EHRPWM_XXXX_XXXX_CLEAR \n`<br>`- EHRPWM_XXXX_XXXX_SET \n`<br>`- EHRPWM_XXXX_XXXX_TOGGLE \n` |

**Returns**

None

#### 4.10.2.12 void EHRPWMConfigureChopperDuty ( unsigned int *baseAddr*, unsigned int *dutyCycle* )

This API configures the chopper duty cyce. In Chopper sub-module the PWM signal is modulated with a carrier signal. Th duty cycle of the carrier signal is configured with this API.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *dutyCycle* | Duty cycle of the chopping carrier. Possible values are :<br>• EHRPWM_DUTY_12_5_PER<br>• EHRPWM_DUTY_25_PER<br>• EHRPWM_DUTY_37_5_PER<br>• EHRPWM_DUTY_50_PER<br>• EHRPWM_DUTY_62_5_PER<br>• EHRPWM_DUTY_75_PER<br>• EHRPWM_DUTY_87_5_PER |

**Returns**

None

#### 4.10.2.13 void EHRPWMConfigureChopperFreq ( unsigned int *baseAddr,* unsigned int *freqDiv* )

This API configures the chopper frequency. In chopper sub-module the PWM signal is modulated with a carrier signal. The frequency of the carrier signal is configured with this API.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *freqDiv* | Frequency divider |

**Returns**

None

#### 4.10.2.14 void EHRPWMConfigureChopperOSPW ( unsigned int *baseAddr,* unsigned int *OSPWCycles* )

This API configures one shot pulse width. The chopper module is useful in switching operations for pulse transformers. The one-shot block provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *OSPWCycles* | Number of clocks the OSPW to be ON. |

**Returns**

None

#### 4.10.2.15 void EHRPWMDBConfigureFED ( unsigned int *baseAddr,* unsigned int *fallingEdgeDelay* )

This API sets the Falling edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *fallingEdgeDelay* | Falling Edge Delay |

**Returns**

None

#### 4.10.2.16 void EHRPWMDBConfigureRED ( unsigned int *baseAddr,* unsigned int *raisingEdgeDelay* )

This API sets the raising edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *raisingEdge↩ Delay* | Raising Edge Delay |

**Returns**

None

#### 4.10.2.17 void EHRPWMDBOutput ( unsigned int *baseAddr,* unsigned int *DBgenOpMode* )

This API selects output mode. This allows to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenOpMode* | Output mode. The possible values can be :<br>• EHRPWM_DBCTL_OUT_MODE_BYPASS<br>• EHRPWM_DBCTL_OUT_MODE_NOREDBFED<br>• EHRPWM_DBCTL_OUT_MODE_AREDNOFED<br>• EHRPWM_DBCTL_OUT_MODE_AREDBFED |

**Returns**

None

#### 4.10.2.18 void EHRPWMDBPolaritySelect ( unsigned int *baseAddr,* unsigned int *DBgenPol* )

This API selects the polarity. This allows to selectively invert one of the delayed signals before it is sent out of the dead-band sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenSource* | Polarity. The possible values can be :<br>• HRPWM_DBCTL_POLSEL_ACTIVEHIGH<br>• EHRPWM_DBCTL_POLSEL_ALC<br>• EHRPWM_DBCTL_POLSEL_AHC<br>• EHRPWM_DBCTL_POLSEL_ACTIVELOW |

**Returns**

None

#### 4.10.2.19 void EHRPWMDBSourceSelect ( unsigned int *baseAddr,* unsigned int *DBgenSource* )

This API selects the source for delay blocks in dead band sub-module. The Dead band generator has two sub-modules, one for raising edge delay and the other for falling edge delay. This can be configured when a delay is need between two signals during signal change. The dead band generator is usefull in full-inverters.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenSource* | Source selection. The possible values can be<br>• EHRPWM_DBCTL_IN_MODE_AREDAFED<br>• EHRPWM_DBCTL_IN_MODE_BREDAFED<br>• EHRPWM_DBCTL_IN_MODE_AREDBFED<br>• EHRPWM_DBCTL_IN_MODE_BREDBFED |

**Returns**

None

#### 4.10.2.20 bool EHRPWMETEventCount ( unsigned int *baseAddr* )

This API returns the number of events occured.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

eventCount number of events occured

#### 4.10.2.21 void EHRPWMETIntClear ( unsigned int *baseAddr* )

This API clears the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.22 void EHRPWMETIntDisable ( unsigned int *baseAddr* )

This API disables the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.23 void EHRPWMETIntEnable ( unsigned int *baseAddr* )

This API enables the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.24 void EHRPWMETIntPrescale ( unsigned int *baseAddr,* unsigned int *prescale* )

This API prescales the event on which interrupt is to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *prescale* | prescalar value |

**Returns**

None

#### 4.10.2.25 void EHRPWMETIntSourceSelect ( unsigned int *baseAddr,* unsigned int *selectInt* )

This API selects the interrupt source.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *selectInt* | Event which triggers interrupt. The possible source can be,<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUZERO<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUPRD<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPAINC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPADEC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBINC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBDEC |

**Returns**

None

#### 4.10.2.26 bool EHRPWMETIntStatus ( unsigned int *baseAddr* )

This API returns the interrupt status.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

status Status of the interrupt

#### 4.10.2.27 void EHRPWMETIntSWForce ( unsigned int *baseAddr* )

This API forces interrupt to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.28 void EHRPWMHRDisable ( unsigned int *baseAddr* )

This API disables the HR sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.29 bool EHRPWMLoadCMPA ( unsigned int *baseAddr,* unsigned int *CMPAVal,* bool *enableShadowWrite,* unsigned int *ShadowToActiveLoadTrigger,* bool *OverwriteShadowFull* )

This API loads the CMPA value. When CMPA value equals the counter value, then an event is generated both in the up direction and down direction.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPAVal* | CMPA value to be loaded. |
| *enableShadow↩ Write* | Enable write to shadow register. |

| | |
|---|---|
| *ShadowToActiveLoadTrigger* | Shadow to active register load trigger. |
| *OverwriteShadowFull* | Overwrite even if previous value is not loaded to active register. |

**Returns**

bool Flag indicates whether the CMPA value is written or not.

#### 4.10.2.30 void EHRPWMLoadCMPAHR ( unsigned int *baseAddr,* unsigned int *CMPAHRVal,* unsigned int *ShadowToActiveLoadTrigger* )

This API loads CMPAHR value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPAHRVal* | CMPAHR value to he loaded |
| *ShadowToActiveLoadTrigger* | Condition when the active reg to be loaded from shadow register. |

**Returns**

None

#### 4.10.2.31 bool EHRPWMLoadCMPB ( unsigned int *baseAddr,* unsigned int *CMPBVal,* bool *enableShadowWrite,* unsigned int *ShadowToActiveLoadTrigger,* bool *OverwriteShadowFull* )

This API loads the CMPB value. When CMPB value equals the counter value, then an event is generated both in the up direction and down direction.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPBVal* | CMPB value to be loaded. |
| *enableShadowWrite* | Enable write to shadow register. |
| *ShadowToActiveLoadTrigger* | Shadow to active register load trigger. |
| *OverwriteShadowFull* | Overwrite even if previous value is not loaded to active register. |

**Returns**

bool Flag indicates whether the CMPB value is written or not.

#### 4.10.2.32 void EHRPWMLoadTBPHSHR ( unsigned int *baseAddr,* unsigned int *TBPHSHRVal* )

This API loads the HR PHS value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *TBPHSHRVal* | TB PHS HR value |

**Returns**

None

#### 4.10.2.33 void EHRPWMPWMOpFreqSet ( unsigned int *baseAddr,* unsigned int *tbClk,* unsigned int *pwmFreq,* unsigned int *counterDir,* bool *enableShadowWrite* )

This API configures the PWM Frequency/Period. The period count determines the period of the final output waveform. For the given period count, in the case of UP and DOWN counter the count value will be loaded as is. In the case of UP_DOWN counter the count is halfed.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbClk* | Timebase clock. |
| *pwmFreq* | Frequency of the PWM Output. If the counter direction is up-down this value has to be halfed, so that the period of the final output is equal to pwmFreq. |
| *counterDir* | Direction of the counter(up, down, up-down) |
| *enableShadowWrite* | Whether write to Period register is to be shadowed |

**Returns**

None.

#### 4.10.2.34 unsigned int EHRPWMReadTBCount ( unsigned int *baseAddr* )

This API gets the TB counter current value. The count operation is not affected by the read.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

tbCount Current Timebase count value.

#### 4.10.2.35 void EHRPWMSWForceA ( unsigned int *baseAddr* )

This API triggers the SW forced event on A. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.36 void EHRPWMSWForceB ( unsigned int *baseAddr* )

This API triggers the SW forced event on B. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.10.2.37 void EHRPWMSyncOutModeSet ( unsigned int *baseAddr,* unsigned int *syncOutMode* )

This API selects the output sync source. It determines on which of the following event sync-out has to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *syncOutMode* | Sync out mode. Possible values are,<br>• EHRPWM_SYNCOUT_SYNCIN<br>• EHRPWM_SYNCOUT_COUNTER_EQUAL_ZERO<br>• EHRPWM_SYNCOUT_COUNTER_EQUAL_COMPAREB<br>• EHRPWM_SYNCOUT_DISABLE |

**Returns**

None.

#### 4.10.2.38 void EHRPWMTBClearStatus ( unsigned int *baseAddr,* unsigned int *tbStatusMask* )

This API clears the TB status bits indicated by the tbStatusMask parameter.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbStatusMask* | Indicates which status bit need to be cleared.<br>• EHRPWM_TBSTS_CTRMAX<br>• EHRPWM_TBSTS_SYNCI<br>• EHRPWM_TBSTS_CTRDIR |

**Returns**

None

#### 4.10.2.39 void EHRPWMTBEmulationModeSet ( unsigned int *baseAddr,* unsigned int *mode* )

This API configures emulation mode. This setting determines the behaviour of Timebase during emulation (debugging).

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *mode* | Emulation mode. Possible values are,<br>• EHRPWM_STOP_AFTER_NEXT_TB_INCREMENT<br>• EHRPWM_STOP_AFTER_A_COMPLETE_CYCLE<br>• EHRPWM_FREE_RUN |

**Returns**

None.

#### 4.10.2.40 unsigned int EHRPWMTBStatusGet ( unsigned int *baseAddr,* unsigned int *tbStatusMask* )

This API gets the TB status as indicated by the tbStatusMask parameter.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbStatusMask* | Indicates which status is needed.<br>• EHRPWM_TBSTS_CTRMAX<br>– whether the counter has reached the max value<br>• EHRPWM_TBSTS_SYNCI<br>– indicates external sync event has occured<br>• EHRPWM_TBSTS_CTRDIR - gives the counter direction |

**Returns**

tbStatus Requested status is returned. The user need to extract the appropriate bits by shifting.

#### 4.10.2.41 void EHRPWMTimebaseClkConfig ( unsigned int *baseAddr,* unsigned int *tbClk,* unsigned int *moduleClk* )

This API configures the clock divider of the Time base module. The clock divider can be calculated using the equation TBCLK = SYSCLKOUT/(HSPCLKDIV LKDIV)

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbClk* | Timebase clock to be generated. |
| *moduleClk* | Input clock of the PWM module (sysclk2) |

**Returns**

None.

#### 4.10.2.42 void EHRPWMTimebaseSyncDisable ( unsigned int *baseAddr* )

This API disables the synchronization. Even if sync-in event occurs the count value will not be reloaded.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None.

#### 4.10.2.43 void EHRPWMTimebaseSyncEnable ( unsigned int *baseAddr,* unsigned int *tbPhsValue,* unsigned int *phsCountDir* )

This API enables the synchronization. When a sync-in event is generated the couter is reloaded with the new value. After sync the counter will use the new value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbPhsValue* | Phase value to be reloaded after sync |
| *phsCountDir* | Count direction after sync. Possible values are<br>• EHRPWM_COUNT_DOWN_AFTER_SYNC<br>• EHRPWM_COUNT_UP_AFTER_SYNC |

**Returns**

None.

#### 4.10.2.44 void EHRPWMTriggerSWSync ( unsigned int *baseAddr* )

This API generates sw forced sync pulse. This API can be used for testing. When this API is called sync-in will be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None.

#### 4.10.2.45 void EHRPWMTZFlagClear ( unsigned int *baseAddr,* unsigned int *flagToClear* )

This API clears the flag.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *flagToClear* | Status to be cleared. The possible values can be,<br>• EHRPWM_TZCLR_OST<br>• EHRPWM_TZCLR_CBC<br>• EHRPWM_TZCLR_INT |

**Returns**

None

#### 4.10.2.46 unsigned int EHRPWMTZFlagGet ( unsigned int *baseAddr,* unsigned int *flagToRead* )

This API returns the flag status requested.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *flagToRead* | status to be read. The possible values can be,<br>• EHRPWM_TZCLR_OST<br>• EHRPWM_TZCLR_CBC<br>• EHRPWM_TZCLR_INT |

**Returns**

None

#### 4.10.2.47 void EHRPWMTZForceAOnTrip ( unsigned int *baseAddr,* unsigned int *opValue* )

This API configures the o/p on A when a trip event is recognized. The output can be set to high or low or high impedence.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *opValue* | o/p state to be configured |

**Returns**

None

#### 4.10.2.48 void EHRPWMTZForceBOnTrip ( unsigned int *baseAddr,* unsigned int *opValue* )

This API configures the o/p on B when a trip event is recognised. The output can be set to high or low or high impedence.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *opValue* | o/p state to be configured |

**Returns**

None

#### 4.10.2.49 void EHRPWMTZIntDisable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API disables the trip interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | disable OST or CBC |

**Returns**

None

#### 4.10.2.50 void EHRPWMTZIntEnable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables the trip interrupt. When trip event occurs the sub-module can be configured to interrupt CPU.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

| | |
|---|---|
| *osht_CBC* | enable OST or CBC |

**Returns**

None

#### 4.10.2.51 void EHRPWMTZSWFrcEvent ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables to generate SW forced trip.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | generate OST or CBC trip |

**Returns**

None

#### 4.10.2.52 void EHRPWMTZTripEventDisable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API disable the trip event. The trip events will be ignored.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | Disable OST or CBC event |

**Returns**

None

#### 4.10.2.53 void EHRPWMTZTripEventEnable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables the trip event. The trip signals indicates external fault, and the ePWM outputs can be programmed to respond accordingly when faults occur.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | Enable OST or CBC event |

**Returns**

None

#### 4.10.2.54 void EHRPWMWriteTBCount ( unsigned int *baseAddr,* unsigned int *tbCount* )

This API loads the TB counter. The new value is taken immediately.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbCount* | Time base count value to be loaded. |

**Returns**

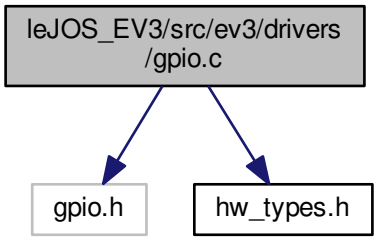
None.

## 4.11 leJOS_EV3/src/ev3/drivers/gpio.c File Reference

This file contains the device abstraction layer APIs for GPIO.

```
#include "gpio.h"
#include "hw_types.h"
```

Include dependency graph for gpio.c:




### Functions

- void GPIODirModeSet (unsigned int baseAdd, unsigned int pinNumber, unsigned int pinDir)

  *This function configures the direction of a pin as input or output.*
- unsigned int GPIODirModeGet (unsigned int baseAdd, unsigned int pinNumber)

  *This function gets the direction of a pin which has been configured as an input or an output pin.*
- void GPIOPinWrite (unsigned int baseAdd, unsigned int pinNumber, unsigned int bitValue)

  *This function writes a logic 1 or a logic 0 to the specified pin.*
- int GPIOPinRead (unsigned int baseAdd, unsigned int pinNumber)

  *This function reads the value(logic level) of an input or an output pin.*
- void GPIOIntTypeSet (unsigned int baseAdd, unsigned int pinNumber, unsigned int intType)

  *This function configures the trigger level type for which an interrupt is required to occur.*
- unsigned int GPIOIntTypeGet (unsigned int baseAdd, unsigned int pinNumber)

  *This function reads the trigger level type being set for interrupts to be generated.*
- unsigned int GPIOPinIntStatus (unsigned int baseAdd, unsigned int pinNumber)

  *This function determines the status of interrupt on a specified pin.*
- void GPIOPinIntClear (unsigned int baseAdd, unsigned int pinNumber)

  *This function clears the interrupt status of the pin being accessed.*
- void GPIOBankIntEnable (unsigned int baseAdd, unsigned int bankNumber)

  *This function enables the interrupt generation capability for the bank of GPIO pins specified.*
- void GPIOBankIntDisable (unsigned int baseAdd, unsigned int bankNumber)

*This function disables the interrupt generation capability for the bank of GPIO pins specified.*

- void GPIOBankPinsWrite (unsigned int baseAdd, unsigned int bankNumber, unsigned int setPins, unsigned int clrPins)

  *This function is used to collectively set and collectively clear the specified bits.*

### 4.11.1 Detailed Description

This file contains the device abstraction layer APIs for GPIO.

### 4.11.2 Function Documentation

#### 4.11.2.1 void GPIOBankIntDisable ( unsigned int *baseAdd,* unsigned int *bankNumber* )

This function disables the interrupt generation capability for the bank of GPIO pins specified.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *bankNumber* | This is the bank for whose pins interrupt generation capability needs to be disabled. bank↩ Number is 0 for bank 0, 1 for bank 1 and so on. |

**Returns**

None.

#### 4.11.2.2 void GPIOBankIntEnable ( unsigned int *baseAdd,* unsigned int *bankNumber* )

This function enables the interrupt generation capability for the bank of GPIO pins specified.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *bankNumber* | This is the bank for whose pins interrupt generation capabiility needs to be enabled. bank↩ Number is 0 for bank 0, 1 for bank 1 and so on. |

**Returns**

None.

#### 4.11.2.3 void GPIOBankPinsWrite ( unsigned int *baseAdd,* unsigned int *bankNumber,* unsigned int *setPins,* unsigned int *clrPins* )

This function is used to collectively set and collectively clear the specified bits.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *bankNumber* | Numerical value of the bank whose pins are to be modified. |
| *setPins* | The bit-mask of the pins whose values have to be set. This could be the bitwise OR of the following macros: -> GPIO_BANK_PIN_n where n >= 0 and n <= 15. |

| | |
|---|---|
| *clrPins* | The bit-mask of the pins whose values have to be cleared. This could be the bitwise OR of the following macros: -> GPIO_BANK_PIN_n where n >= 0 and n <= 15. |

**Returns**

None.

**Note**

The pre-requisite to write to any pins is that the pins have to be configured as output pins.

#### 4.11.2.4 unsigned int GPIODirModeGet ( unsigned int *baseAdd,* unsigned int *pinNumber* )

This function gets the direction of a pin which has been configured as an input or an output pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns one of the following two values: 1> GPIO_DIR_INPUT, if the pin is configured as an input pin. 2> GPIO_DIR_OUTPUT, if the pin is configured as an output pin.

#### 4.11.2.5 void GPIODirModeSet ( unsigned int *baseAdd,* unsigned int *pinNumber,* unsigned int *pinDir* )

This function configures the direction of a pin as input or output.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |
| *pinDir* | The direction to be set for the pin. This can take the values: 1> GPIO_DIR_INPUT, for configuring the pin as input. 2> GPIO_DIR_OUTPUT, for configuring the pin as output. |

**Returns**

None.

**Note**

Here we write to the DIRn register. Writing a logic 1 configures the pin as input and writing logic 0 as output. By default, all the pins are set as input pins.

#### 4.11.2.6 unsigned int GPIOIntTypeGet ( unsigned int *baseAdd,* unsigned int *pinNumber* )

This function reads the trigger level type being set for interrupts to be generated.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin to be accessed. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns a value which indicates the type of trigger level type being set. One of the following values is returned. 1> GPIO_INT_TYPE_NOEDGE, indicating no interrupts will be generated over the corresponding pin. 2> GPIO_INT_TYPE_FALLEDGE, indicating a falling edge on the corresponding pin signifies an interrupt generation. 3> GPIO_INT_TYPE_RISEDGE, indicating a rising edge on the corresponding pin signifies an interrupt generation. 4> GPIO_INT_TYPE_BOTHEDGE, indicating both edges on the corresponding pin signifies an interrupt each being generated.

#### 4.11.2.7 void GPIOIntTypeSet ( unsigned int *baseAdd,* unsigned int *pinNumber,* unsigned int *intType* )

This function configures the trigger level type for which an interrupt is required to occur.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |
| *intType* | This specifies the trigger level type. This can take one of the following four values: 1> GPIO_INT_TYPE_NOEDGE, to not generate any interrupts. 2> GPIO_INT_TYPE_FALLEDGE, to generate an interrupt on the falling edge of a signal on that pin. 3> GPIO_INT_TYPE_RISEDGE, to generate an interrupt on the rising edge of a signal on that pin. 4> GPIO_INT_TYPE_BOTHEDGE, to generate interrupts on both rising and falling edges of a signal on that pin. |

**Returns**

None.

**Note**

Configuring the trigger level type for generating interrupts is not enough for the GPIO module to generate interrupts. The user should also enable the interrupt generation capability for the bank to which the pin belongs to. Use the function GPIOBankIntEnable() to do the same.

#### 4.11.2.8 void GPIOPinIntClear ( unsigned int *baseAdd,* unsigned int *pinNumber* )

This function clears the interrupt status of the pin being accessed.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin to be accessed. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

None.

#### 4.11.2.9 unsigned int GPIOPinIntStatus ( unsigned int *baseAdd,* unsigned int *pinNumber* )

This function determines the status of interrupt on a specified pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin to be accessed. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns a value which expresses the status of an interrupt raised over the specified pin. 1> GPIO_INT↩ _NOPEND, if no interrupts are left to be serviced. 2> GPIO_INT_PEND, if the interrupt raised over that pin is yet to be cleared and serviced.

**Note**

If an interrupt over a pin is found to be pending, then the application can call GPIOPinIntClear() to clear the interrupt status.

#### 4.11.2.10 int GPIOPinRead ( unsigned int *baseAdd,* unsigned int *pinNumber* )

This function reads the value(logic level) of an input or an output pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns the value present on the specified pin. This returns one of the following values: 1> GPIO_PIN↩ _LOW, if the value on the pin is logic 0. 2> GPIO_PIN_HIGH, if the value on the pin is logic 1.

**Note**

Using this function, we can read the values of both input and output pins.

#### 4.11.2.11 void GPIOPinWrite ( unsigned int *baseAdd,* unsigned int *pinNumber,* unsigned int *bitValue* )

This function writes a logic 1 or a logic 0 to the specified pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |
| *bitValue* | This signifies whether to write a logic 0 or logic 1 to the specified pin.This variable can take any of the following two values: 1> GPIO_PIN_LOW, which indicates to clear(logic 0) the bit. 2> GPIO_PIN_HIGH, which indicates to set(logic 1) the bit. |

**Returns**

None.

**Note**

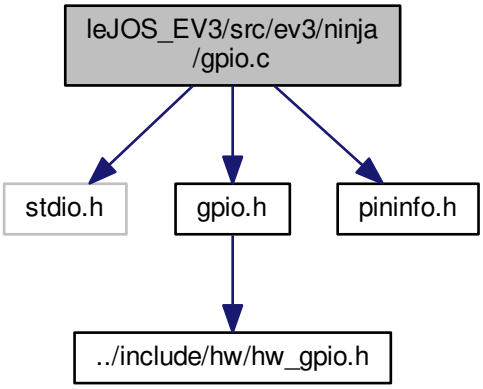
The pre-requisite to write to any pin is that the pin has to be configured as an output pin.

## 4.12 leJOS_EV3/src/ev3/ninja/gpio.c File Reference

This contains function definitions required to interact with GPIO pins of the SoC.

```
#include "stdio.h"
#include "gpio.h"
#include "pininfo.h"
```

Include dependency graph for gpio.c:




### Macros

- #define SYSCFG_KICK(N) (∗((volatile unsigned int∗)(SYSCFG0_BASE + 0x38 + (N) ∗ 0x4)))

  *The KICK registers required to unlock access to some SYSCFG register (including the pin-multiplexing registers)*
- #define SYSCFG_PINMUX(N) (∗((volatile unsigned int∗)(SYSCFG0_BASE + 0x120 + (N) ∗ 0x4)))

  *The PINMUX registers in the SYSCFG0 register in order to manipulate the pin-multiplexing configuration.*
- #define KICK0_UNLOCK 0x83E70B13

  *The value which needs to be written to KICK0 register in order to unlock the protected registers.*
- #define KICK1_UNLOCK 0x95A4F1E0

  *The value which needs to be written to KICK1 register in order to unlock the protected registers.*
- #define KICK0_LOCK 0x0

  *Any random value which can to be written to KICK0 register in order to lock the protected registers.*
- #define KICK1_LOCK 0x0

  *Any random value which can to be written to KICK1 register in order to lock the protected registers.*
- #define SYSCFG_UNLOCK { SYSCFG_KICK(0) = KICK0_UNLOCK; SYSCFG_KICK(1) = KICK1_UNLOCK; }

  *Unlock the protected registers by writing the required values into the KICK registers.*
- #define SYSCFG_LOCK { SYSCFG_KICK(0) = KICK0_LOCK; SYSCFG_KICK(1) = KICK1_LOCK; }

  *Lock the protected registers by writing random values into the KICK registers.*

### Functions

- void gpio_init_pin (unsigned int pin)

  *Initialize a GPIO pin with its GPIO functionality.*

- void gpio_init_outpin (unsigned int pin)

  *Initialize a GPIO pin with its GPIO functionality and set its direction to output.*
- void gpio_init_inpin (unsigned int pin)

  *Initialize a GPIO pin with its GPIO functionality and set its direction to input.*
- void gpio_set (unsigned int pin, unsigned int value)

  *Set the value of a GPIO pin which is configured as output.*
- unsigned int gpio_get (unsigned int pin)

  *Get the signal at a GPIO pin which is configured as input.*
- void turn_off_brick (void)

  *Turn the EV3 off by setting GPIO pin 6 on bank 11 as an output pin.*

### 4.12.1 Detailed Description

This contains function definitions required to interact with GPIO pins of the SoC.

**Author**

Tobias Schießl, ev3ninja

### 4.12.2 Function Documentation

#### 4.12.2.1 unsigned int gpio_get ( unsigned int *pin* )

Get the signal at a GPIO pin which is configured as input.

If not already done, this function will set the pin as an input pin.

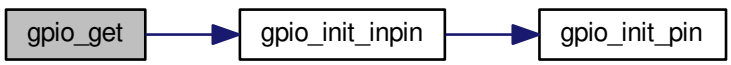
**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

The signal at the pin: 1 for high and 0 for low

Here is the call graph for this function:




#### 4.12.2.2 void gpio_init_inpin ( unsigned int *pin* )

Initialize a GPIO pin with its GPIO functionality and set its direction to input.

**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

none

Here is the call graph for this function:


#### 4.12.2.3 void gpio_init_outpin ( unsigned int *pin* )

Initialize a GPIO pin with its GPIO functionality and set its direction to output.

**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

none

Here is the call graph for this function:


#### 4.12.2.4 void gpio_init_pin ( unsigned int *pin* )

Initialize a GPIO pin with its GPIO functionality.

**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

none

#### 4.12.2.5 void gpio_set ( unsigned int *pin,* unsigned int *value* )

Set the value of a GPIO pin which is configured as output.

If not already done, this function will set the pin as an output pin.

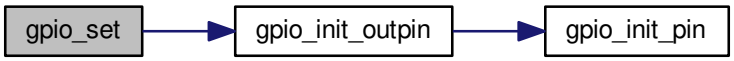
**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |
| *value* | - The value to set (0 will be low, all other values will map to high) |

**Returns**

none

Here is the call graph for this function:



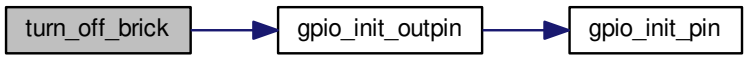

#### 4.12.2.6 void turn_off_brick ( void )

Turn the EV3 off by setting GPIO pin 6 on bank 11 as an output pin.

**Returns**

none

Here is the call graph for this function:

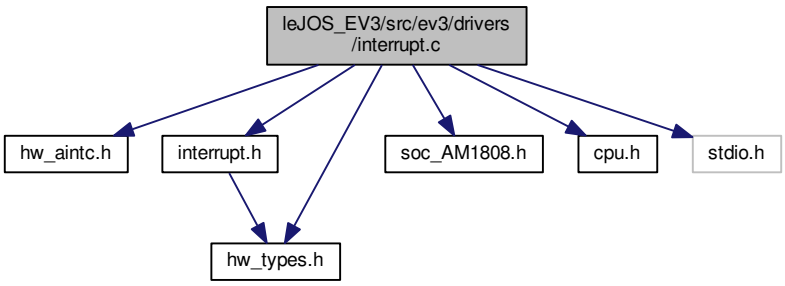



## 4.13 leJOS_EV3/src/ev3/drivers/interrupt.c File Reference

Contains the APIs for configuring AINTC.

```
#include "hw_aintc.h"
#include "interrupt.h"
#include "hw_types.h"
#include "soc_AM1808.h"
#include "cpu.h"
#include <stdio.h>
```

Include dependency graph for interrupt.c:




## Macros

- #define **SYSTEM_INTR_BYTE_SHIFT** (0x03)
- #define **SYSTEM_INTR_BIT_MASK** (0x07)

## Functions

- void IntRegister (unsigned int intrNum, void(*fnHandler)(void))

  *Registers an interrupt Handler in the interrupt vector table for system interrupts.*
- void IntUnRegister (unsigned int intrNum)

  *Unregisters an interrupt.*
- void IntChannelSet (unsigned int intrNum, unsigned char channel)

  *Set the channel number for a system interrupt. Channel numbers 0-1 are mapped to FIQ and Channel numbers 2-31 are mapped to IRQ of ARM. One or more system interrupt can be mapped to one channel. However, one system interrupt can not be mapped to multiple channels.*
- unsigned char IntChannelGet (unsigned int intrNum)

  *Get the channel number for a system interrupt.*
- void IntGlobalEnable (void)

  *Enables interrupts in GER register of AINTC. FIQ and IRQ will be signaled for processing.*
- void IntGlobalDisable (void)

  *Disables interrupts in GER register of AINTC. No interrupts will be signaled by the AINTC to the ARM core.*
- void IntSystemEnable (unsigned int intrNum)

  *Enables the sytem interrupt in AINTC. The interrupt will be processed only if it is enabled in AINTC.*
- void IntSystemDisable (unsigned int intrNum)

  *Disables the sytem interrupt in the AINTC.*
- void IntSystemStatusClear (unsigned int intrNum)

  *Clears a sytem interrupt status in AINTC.*
- unsigned int IntSystemStatusRawGet (unsigned int intrNum)

  *Get the raw status of a system interrupt. This will return 1 if the status is set and return 0 if the status is clear.*
- unsigned int IntSystemStatusEnabledGet (unsigned int intrNum)

*Get the enabled status of a system interrupt. This will return 1 if the status is set, and return 0 if the status is clear.*

- void IntIRQEnable (void)

  *Enables IRQ in HIER register of AINTC.*
- void IntIRQDisable (void)

  *Disables IRQ in HIER register of AINTC.*
- void IntFIQEnable (void)

  *Enables FIQ in AINTC.*
- void IntFIQDisable (void)

  *Disables FIQ host interrupts in AINTC.*
- void IntAINTCInit (void)

  *This API is used to setup the AINTC. This API shall be called before using the AINTC. All the host interruptsi will be disabled after calling this API. The user shall enable all the interrupts required for processing.*
- void IntMasterIRQEnable (void)

  *Enables the processor IRQ only in CPSR. Makes the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- void IntMasterIRQDisable (void)

  *Disables the processor IRQ only in CPSR.Prevents the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- void IntMasterFIQEnable (void)

  *Enables the processor FIQ only in CPSR. Makes the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- void IntMasterFIQDisable (void)

  *Disables the processor FIQ only in CPSR.Prevents the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- unsigned int IntMasterStatusGet (void)

  *Returns the status of the interrupts FIQ and IRQ.*
- unsigned char IntDisable (void)

  *Read and save the stasus and Disables the processor IRQ . Prevents the processor to respond to IRQs.*
- void IntEnable (unsigned char status)

  *Restore the processor IRQ only status. This does not affect the set of interrupts enabled/disabled in the AINTC.*

### 4.13.1 Detailed Description

Contains the APIs for configuring AINTC.

### 4.13.2 Function Documentation

#### 4.13.2.1 void IntAINTCInit ( void )

This API is used to setup the AINTC. This API shall be called before using the AINTC. All the host interruptsi will be disabled after calling this API. The user shall enable all the interrupts required for processing.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.13.2.2 unsigned char IntChannelGet ( unsigned int *intrNum* )

Get the channel number for a system interrupt.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |

**Returns**

Channel Number.

#### 4.13.2.3 void IntChannelSet ( unsigned int *intrNum,* unsigned char *channel* )

Set the channel number for a system interrupt. Channel numbers 0-1 are mapped to FIQ and Channel numbers 2-31 are mapped to IRQ of ARM. One or more system interrupt can be mapped to one channel. However, one system interrupt can not be mapped to multiple channels.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |
| *channel* | - Channel Number to be set |

**Returns**

None.

#### 4.13.2.4 unsigned char IntDisable ( void )

Read and save the stasus and Disables the processor IRQ . Prevents the processor to respond to IRQs.

**Parameters**

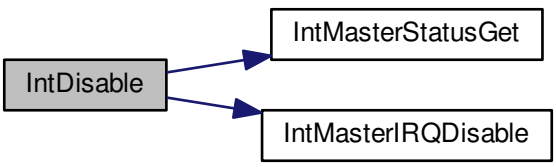
| | |
|---|---|
| *None* | |

**Returns**

Current status of IRQ

Note: This function call shall be done only in previleged mode of ARM

Here is the call graph for this function:




#### 4.13.2.5 void IntEnable ( unsigned char *status* )

Restore the processor IRQ only status. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *The* | status returned by the IntDisable fundtion. |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

Here is the call graph for this function:


#### 4.13.2.6 void IntFIQDisable ( void )

Disables FIQ host interrupts in AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

#### 4.13.2.7 void IntFIQEnable ( void )

Enables FIQ in AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.13.2.8 void IntGlobalDisable ( void )

Disables interrupts in GER register of AINTC. No interrupts will be signaled by the AINTC to the ARM core.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

#### 4.13.2.9 void IntGlobalEnable ( void )

Enables interrupts in GER register of AINTC. FIQ and IRQ will be signaled for processing.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

#### 4.13.2.10 void IntIRQDisable ( void )

Disables IRQ in HIER register of AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.13.2.11 void IntIRQEnable ( void )

Enables IRQ in HIER register of AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.13.2.12 void IntMasterFIQDisable ( void )

Disables the processor FIQ only in CPSR.Prevents the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.13.2.13 void IntMasterFIQEnable ( void )

Enables the processor FIQ only in CPSR. Makes the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.13.2.14 void IntMasterIRQDisable ( void )

Disables the processor IRQ only in CPSR.Prevents the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.13.2.15 void IntMasterIRQEnable ( void )

Enables the processor IRQ only in CPSR. Makes the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.13.2.16 unsigned int IntMasterStatusGet ( void )

Returns the status of the interrupts FIQ and IRQ.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

Status of interrupt as in CPSR.

Note: This function call shall be done only in previleged mode of ARM

#### 4.13.2.17 void IntRegister ( unsigned int *intrNum,* void(*)(void) *fnHandler* )

Registers an interrupt Handler in the interrupt vector table for system interrupts.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |
| *fnHandler* | - Function pointer to the ISR |

Note: When the interrupt occurs for the sytem interrupt number indicated, the control goes to the ISR given as the parameter.

**Returns**

None.

#### 4.13.2.18 void IntSystemDisable ( unsigned int *intrNum* )

Disables the sytem interrupt in the AINTC.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

None

#### 4.13.2.19 void IntSystemEnable ( unsigned int *intrNum* )

Enables the sytem interrupt in AINTC. The interrupt will be processed only if it is enabled in AINTC.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

None.

#### 4.13.2.20 void IntSystemStatusClear ( unsigned int *intrNum* )

Clears a sytem interrupt status in AINTC.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

None

#### 4.13.2.21 unsigned int IntSystemStatusEnabledGet ( unsigned int *intrNum* )

Get the enabled status of a system interrupt. This will return 1 if the status is set, and return 0 if the status is clear.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |

**Returns**

Enabled Interrupt Status.

#### 4.13.2.22 unsigned int IntSystemStatusRawGet ( unsigned int *intrNum* )

Get the raw status of a system interrupt. This will return 1 if the status is set and return 0 if the status is clear.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

Raw Interrupt Status

#### 4.13.2.23 void IntUnRegister ( unsigned int *intrNum* )

Unregisters an interrupt.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |

Note: Once an interrupt is unregistered it will enter infinite loop once an interrupt occurs

**Returns**

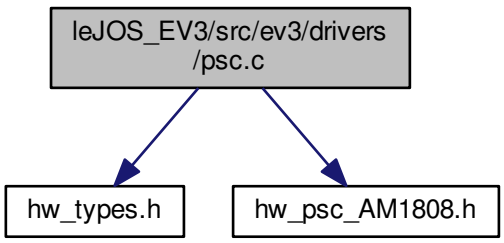
None.

## 4.14 leJOS_EV3/src/ev3/drivers/psc.c File Reference

This file contains the device abstraction layer APIs for the PSC module. There are APIs here to enable power domain, transitions for a particular module.

```
#include "hw_types.h"
#include "hw_psc_AM1808.h"
```

Include dependency graph for psc.c:



### Functions

- int PSCModuleControl (unsigned int baseAdd, unsigned int moduleId, unsigned int powerDomain, unsigned int flags)

  *This function sets the requested module in the required state.*

### 4.14.1 Detailed Description

This file contains the device abstraction layer APIs for the PSC module. There are APIs here to enable power domain, transitions for a particular module.

### 4.14.2 Function Documentation

#### 4.14.2.1 int PSCModuleControl ( unsigned int *baseAdd,* unsigned int *moduleId,* unsigned int *powerDomain,* unsigned int *flags* )

This function sets the requested module in the required state.

**Parameters**

| | |
|---|---|
| *baseAdd* | Memory address of the PSC instance used. |
| *moduleId* | The module number of the module to be commanded. |
| *powerDomain* | The power domain of the module to be commanded. |
| *flags* | This contains the flags that is a logical OR of the commands that can be given to a module. |

**Returns**

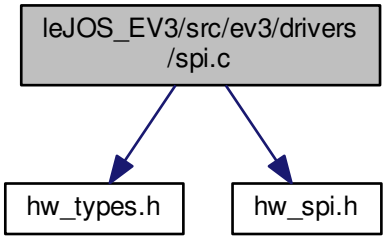
0 in case of successful transition, -1 otherwise.

## 4.15 leJOS_EV3/src/ev3/drivers/spi.c File Reference

SPI device abstraction layer APIs.

```
#include "hw_types.h"
#include "hw_spi.h"
```

Include dependency graph for spi.c:



## Functions

- void SPIClkConfigure (unsigned int baseAdd, unsigned int moduleClk, unsigned int spiClk, unsigned int dataFormat)
  - *It will configure the prescalar to generate required spi clock*
  .
- void SPIEnable (unsigned int baseAdd)
  - *It will Enable SPI module*
  .
- void SPIReset (unsigned int baseAdd)
  - *It will put SPI in to reset state.*
- void SPIOutOfReset (unsigned int baseAdd)
  - *Brings SPI out of reset state.*
- void SPIModeConfigure (unsigned int baseAdd, unsigned int flag)
  - *Configures SPI to master or slave mode.*
- void SPIPinControl (unsigned int baseAdd, unsigned int idx, unsigned int flag, unsigned int *val)
  - *Configures SPI Pin Control Registers.*
- void SPIDelayConfigure (unsigned int baseAdd, unsigned int c2edelay, unsigned int t2edelay, unsigned int t2cdelay, unsigned int c2tdelay)
  - *Configures SPI CS and ENA Delay in SPIDELAY Register.*
- void SPIDefaultCSSet (unsigned int baseAdd, unsigned char dcsval)
  - *Sets the default value for CS pin(line) when no transmission is is performed by writing to SPIDEF Reg.*
- void SPIDat1Config (unsigned int baseAdd, unsigned int flag, unsigned char cs)
  - *Configures the SPIDat1 Register.It won't write to TX part of the SPIDat1 Register.*
- void SPITransmitData1 (unsigned int baseAdd, unsigned int data)
  - *Trasmits Data from TX part of SPIDAT1 register.*
- void SPICSTimerDisable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Disables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).*
- void SPICSTimerEnable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Enables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).*
- void SPICharLengthSet (unsigned int baseAdd, unsigned int numOfChar, unsigned int dataFormat)
  - *It Set the Charcter length.*
- void SPIConfigClkFormat (unsigned int baseAdd, unsigned int flag, unsigned int dataFormat)
  - *It Configures SPI Clock's Phase and Polarity.*
- void SPIShiftMsbFirst (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI to Transmit MSB bit first during Data transfer.*
- void SPIShiftLsbFirst (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI to Transmit LSB bit first during Data transfer.*
- void SPIParityEnable (unsigned int baseAdd, unsigned int flag, unsigned int dataFormat)
  - *It Enables Parity in SPI and also configures Even or Odd Parity.*
- void SPIParityDisable (unsigned int baseAdd, unsigned int dataFormat)

  - *It Disables Parity in SPI.*

- void SPIWdelaySet (unsigned int baseAdd, unsigned int flag, unsigned int dataFormat)
  - *It sets the Delay between SPI transmission.*

- void SPIWaitEnable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI Master to wait for SPIx_ENA signal.*

- void SPIWaitDisable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI Master not to wait for SPIx_ENA signal.*

- void SPIIntLevelSet (unsigned int baseAdd, unsigned int flag)
  - *It Configures SPI to Map interrupts to interrupt line INT1.*

- void SPIIntEnable (unsigned int baseAdd, unsigned int flag)
  - *It Enables the interrupts.*
- void SPIIntDisable (unsigned int baseAdd, unsigned int flag)
  - *It Disables the interrupts.*
- void SPIIntStatusClear (unsigned int baseAdd, unsigned int flag)
  - *It clears Status of interrupts.*
- unsigned int SPIIntStatus (unsigned int baseAdd, unsigned int flag)
  - *It reads the Status of interrupts.*
- unsigned int SPIDataReceive (unsigned int baseAdd)
  - *It recevies data by reading from SPIBUF register.*

- unsigned int SPIInterruptVectorGet (unsigned int baseAdd)
  - *It returns the vector of the pending interrupt at interrupt line INT1.*

### 4.15.1 Detailed Description

SPI device abstraction layer APIs.

### 4.15.2 Function Documentation

#### 4.15.2.1 void SPICharLengthSet ( unsigned int *baseAdd,* unsigned int *numOfChar,* unsigned int *dataFormat* )

- It Set the Charcter length.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value which determines the number of the charcters to be transmitted . |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.15.2.2 void SPIClkConfigure ( unsigned int *baseAdd,* unsigned int *moduleClk,* unsigned int *spiClk,* unsigned int *dataFormat* )

- It will configure the prescalar to generate required spi clock

.

**Parameters**

| | |
|---|---|
| - | baseAdd is Memory Address of the SPI instance used<br>. |
| - | moduleClk is the input clk to SPI module from PLL<br>. |
| - | spiClk is the spi bus speed<br>. |
| - | dataFormat is instance of the data format register used<br>. |

**Returns**

none.

#### 4.15.2.3 void SPIConfigClkFormat ( unsigned int *baseAdd,* unsigned int *flag,* unsigned int *dataFormat* )

- It Configures SPI Clock's Phase and Polarity.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value which determines Phase and Polarity of the Clock..<br>flag can take following values.<br>SPI_CLK_POL_HIGH - Clock is High Before and after data transfer. SPI_CLK_POL_LOW - Clock is Low Before and after data transfer. SPI_CLK_INPHASE - Clock is not Delayed. SPI_CLK_OUTOFPHASE - Clock is Delayed by half clock cycle. |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.15.2.4 void SPICSTimerDisable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Disables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.15.2.5 void SPICSTimerEnable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Enables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.15.2.6 void SPIDat1Config ( unsigned int *baseAdd,* unsigned int *flag,* unsigned char *cs* )

- Configures the SPIDat1 Register.It won't write to TX part of the SPIDat1 Register.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI data instance used. |
| - | flag is value to Configure CSHOL,Wait Delay Conter Enable bit and to select the appropriate DataFormat register.<br>flag can take following values.<br>SPI_CSHOLD - To Hold the CS line active after data Transfer untill new data and Control information is loaded.<br>SPI_DELAY_COUNTER_ENA - Enables Delay Counter.<br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3. |
| - | cs is the value to driven on CS pin(line). |

**Returns**

none.

#### 4.15.2.7 unsigned int SPIDataReceive ( unsigned int *baseAdd* )

- It recevies data by reading from SPIBUF register.

**Parameters**

| | |
|---|---|
| - | baseAdd is the memory instance to be used. |

**Returns**

received data.

#### 4.15.2.8 void SPIDefaultCSSet ( unsigned int *baseAdd,* unsigned char *dcsval* )

- Sets the default value for CS pin(line) when no transmission is is performed by writing to SPIDEF Reg.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | dcsval is the value written to SPIDEF register to set default value on CS pin(line). |

**Returns**

none.

#### 4.15.2.9 void SPIDelayConfigure ( unsigned int *baseAdd,* unsigned int *c2edelay,* unsigned int *t2edelay,* unsigned int *t2cdelay,* unsigned int *c2tdelay* )

- Configures SPI CS and ENA Delay in SPIDELAY Register.

**Parameters**

| | |
|---|---|
| - | baseAdd is Memory Address of the SPI instance. |
| - | c2edelay is the Chip-select-active-to-SPIx_ENA-signal -active-time-out. |
| - | t2edelay is the Transmit-data-finished-to-SPIx_ENA-pin -inactive-time-out. |
| - | t2cdelay is the Transmit-end-to-chip-select-inactive-delay. |
| - | c2tdelay is the Chip-select-active-to-transmit-start-delay. |

**Returns**

none.

Note: SPIx_CS and SPI_ENA are active low pins.

#### 4.15.2.10 void SPIEnable ( unsigned int *baseAdd* )

- It will Enable SPI module
  .

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used<br>. |

**Returns**

none.

#### 4.15.2.11 void SPIIntDisable ( unsigned int *baseAdd,* unsigned int *flag* )

- It Disables the interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts required to be Enabled.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n |

**Returns**

none.

#### 4.15.2.12 void SPIIntEnable ( unsigned int *baseAdd,* unsigned int *flag* )

- It Enables the interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts required to be Enabled.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n |

**Returns**

none.

#### 4.15.2.13 unsigned int SPIInterruptVectorGet ( unsigned int *baseAdd* )

- It returns the vector of the pending interrupt at interrupt line INT1.

**Parameters**

| | |
|---|---|
| - | baseAdd is the memory instance to be used. |

**Returns**

vector of the pending interrupt at interrupt line INT1.

#### 4.15.2.14 void SPIIntLevelSet ( unsigned int *baseAdd,* unsigned int *flag* )

- It Configures SPI to Map interrupts to interrupt line INT1.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts required to be mapped.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INTLVL – Data length error interrupt level.\n<br>SPI_TIMEOUT_INTLVL – TimeOut length error interrupt level.\n<br>SPI_PARITY_ERR_INTLVL – Parity error interrupt level.\n<br>SPI_DESYNC_SLAVE_INTLVL – Desyncrozied slave interrupt level.\n<br>SPI_BIT_ERR_INTLVL – Bit error interrupt level.\n<br>SPI_RECV_OVERRUN_INTLVL – Receive Overrun interrupt level.\n<br>SPI_RECV_INTLVL – Receive interrupt level.\n<br>SPI_TRANSMIT_INTLVL – Transmit interrupt level.\n |

**Returns**

none.

#### 4.15.2.15 unsigned int SPIIntStatus ( unsigned int *baseAdd,* unsigned int *flag* )

- It reads the Status of interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts whose status needs to be read.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n<br>SPI_DMA_REQUEST_ENA_INT – DMA request interrupt.\n |

**Returns**

status of interrupt.

#### 4.15.2.16 void SPIIntStatusClear ( unsigned int *baseAdd,* unsigned int *flag* )

- It clears Status of interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts whose status needs to be cleared.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n<br>SPI_DMA_REQUEST_ENA_INT – DMA request interrupt.\n |

**Returns**

none.

#### 4.15.2.17 void SPIModeConfigure ( unsigned int *baseAdd,* unsigned int *flag* )

- Configures SPI to master or slave mode.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |

**Returns**

none.

#### 4.15.2.18 void SPIOutOfReset ( unsigned int *baseAdd* )

- Brings SPI out of reset state.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |

**Returns**

none.

#### 4.15.2.19 void SPIParityDisable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Disables Parity in SPI.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.15.2.20 void SPIParityEnable ( unsigned int *baseAdd,* unsigned int *flag,* unsigned int *dataFormat* )

- It Enables Parity in SPI and also configures Even or Odd Parity.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value determines whether odd or even Parity.<br><br>flag can take following values.\n<br><br>SPI_ODD_PARITY - selects odd parity<br>SPI_EVEN_PARITY - selects even parity |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.15.2.21 void SPIPinControl ( unsigned int *baseAdd,* unsigned int *idx,* unsigned int *flag,* unsigned int * *val* )

- Configures SPI Pin Control Registers.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | idx is the Pin Control register number.It can take any integer value between 0 and 5. |
| - | flag is to indicate to whether to (1)read from Pin Control Register or to (0)write to Pin Control Register. |
| - | val is a value/return argument which has the value to be written in case of writes or the value read in case of reads |

**Returns**

none.

#### 4.15.2.22 void SPIReset ( unsigned int *baseAdd* )

- It will put SPI in to reset state.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |

**Returns**

none.

#### 4.15.2.23 void SPIShiftLsbFirst ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI to Transmit LSB bit first during Data transfer.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.15.2.24 void SPIShiftMsbFirst ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI to Transmit MSB bit first during Data transfer.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.15.2.25 void SPITransmitData1 ( unsigned int *baseAdd,* unsigned int *data* )

- Trasmits Data from TX part of SPIDAT1 register.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the SPI instance used. |
| - | data is the data transmitted out of SPI. |

**Returns**

none.

#### 4.15.2.26 void SPIWaitDisable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI Master not to wait for SPIx_ENA signal.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

Note:It is applicable only in SPI Master Mode.SPIx_ENA is a active low signal

#### 4.15.2.27 void SPIWaitEnable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI Master to wait for SPIx_ENA signal.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

Note:It is applicable only in SPI Master Mode.SPIx_ENA is a active low signal

#### 4.15.2.28 void SPIWdelaySet ( unsigned int *baseAdd,* unsigned int *flag,* unsigned int *dataFormat* )

- It sets the Delay between SPI transmission.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value determines amount of delay. |

| | |
|---|---|
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

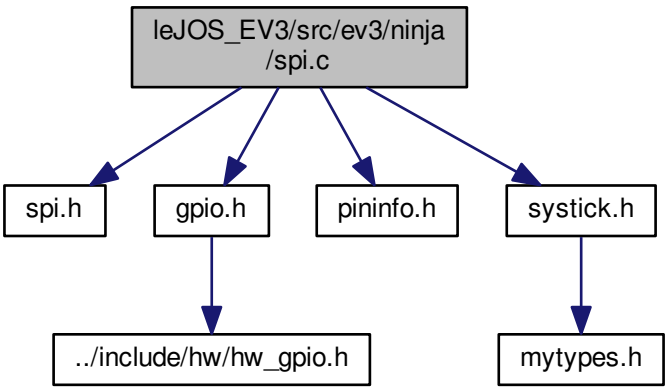
none.

## 4.16 leJOS_EV3/src/ev3/ninja/spi.c File Reference

This contains function definitions required to interact with the SPI0 controller of the SoC which is connected to the ADC.

```
#include "spi.h"
#include "gpio.h"
#include "pininfo.h"
#include "systick.h"
```

Include dependency graph for spi.c:




### Macros

- #define SPI0_CLOCK 150000000UL

  *The clock rate at which the SPI0 controller should run.*
- #define ADC_TIME 8UL

  *The delay between 2 ADC convertions.*
- #define ADC_CLOCK ((1000000UL * 16UL) / ADC_TIME)

  *The clock rate at which the ADC runs.*
- #define CNVSPD ((SPI0_CLOCK / ADC_CLOCK) - 1)

  *The maximum convertion speed.*

- #define SPI0_OFFSET (GPIO_PIN(9, 0) + 5)

  *The offset of the first GPIO pin which can provide SPI functionality.*
- #define SPI0_MOSI (SPI0_OFFSET + 0)

  *The GPIO pin which provided the Master-Output-Slave-Input functionality for SPI0.*
- #define SPI0_MISO (SPI0_OFFSET + 1)

  *The GPIO pin which provided the Master-Input-Slave-Output functionality for SPI0.*
- #define SPI0_SCL (SPI0_OFFSET + 2)

  *The GPIO pin which provided the Clock functionality for SPI0.*
- #define SPI0_CS (SPI0_OFFSET + 3)

  *The GPIO pin which provided the Chip Select functionality for SPI0.*
- #define SPI_BASE ((volatile void*)0x01C41000)

  *The base address of the SPI0 control register.*
- #define SPIGCR0 (*((volatile unsigned int*)(SPI_BASE + 0x00)))

  *The SPI0 global configuration register 0.*
- #define SPIGCR1 (*((volatile unsigned int*)(SPI_BASE + 0x04)))

  *The SPI0 global configuration register 1.*
- #define SPIINT0 (*((volatile unsigned int*)(SPI_BASE + 0x08)))

  *The SPI0 interrupt register.*
- #define SPIPC0 (*((volatile unsigned int*)(SPI_BASE + 0x14)))

  *The SPI0 pin control register 0.*
- #define SPIDAT0 (*((volatile unsigned int*)(SPI_BASE + 0x38)))

  *The SPI0 data transmit register 0.*
- #define SPIDAT1 (*((volatile unsigned int*)(SPI_BASE + 0x3C)))

  *The SPI0 data transmit register 1.*
- #define SPIBUF (*((volatile unsigned int*)(SPI_BASE + 0x40)))

  *The SPI0 receive buffer register.*
- #define SPIDELAY (*((volatile unsigned int*)(SPI_BASE + 0x48)))

  *The SPI0 delay register.*
- #define SPIDEF (*((volatile unsigned int*)(SPI_BASE + 0x4C)))

  *The SPI0 default chip select register.*
- #define SPIFMT0 (*((volatile unsigned int*)(SPI_BASE + 0x50)))

  *The SPI0 data format register 0.*
- #define SPITxFULL (SPIBUF & 0x20000000)

  *Check if the transmit data buffer is full.*
- #define SPIRxEMPTY (SPIBUF & 0x80000000)

  *Check if the receive data buffer is empty.*

## Functions

- unsigned short spi_update (unsigned short data)

  *Send data via the SPI0 controller and get an updated value received by the SPI0 controller.*
- void spi_init (void)

  *Initialize the SPI0 controller and the required GPIO pins.*

### 4.16.1 Detailed Description

This contains function definitions required to interact with the SPI0 controller of the SoC which is connected to the ADC.

**Author**

Tobias Schießl, ev3ninja

### 4.16.2 Function Documentation

#### 4.16.2.1 void spi_init ( void )

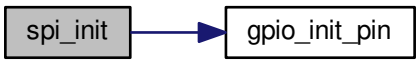
Initialize the SPI0 controller and the required GPIO pins.

For the GPIO pins, the SPI functionality will be set instead of the GPIO functionality.

**Returns**

none

Here is the call graph for this function:




#### 4.16.2.2 unsigned short spi_update ( unsigned short *data* )

Send data via the SPI0 controller and get an updated value received by the SPI0 controller.

Since the SPI0 controller is configured for chip select 3 only, all data sent and received will be to or from the ADC. According to the documentation of the ADC, the selected channel (0 to 15) will be probed and returned on frame n+2. Therefore, the command is sent 3 times and only the third value received is returned to the caller of this function.

**Parameters**

| | |
|---|---|
| *data* | - The data to send (since this is data to the ADC, check the ADC documentation for possible commands that can be sent) |

**Returns**

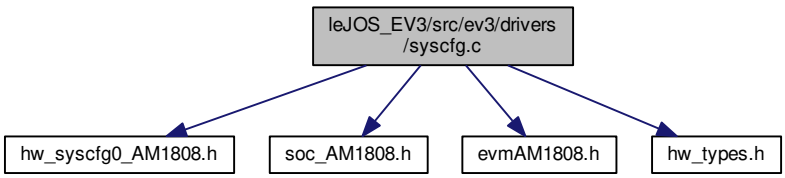
The value received from the ADC after sending the data 3 times

## 4.17 leJOS_EV3/src/ev3/drivers/syscfg.c File Reference

This file contains APIs to lock and unlock the System Configuration (SYSCFG) module registers by appropriately programming the Kick Registers.

```
#include "hw_syscfg0_AM1808.h"
#include "soc_AM1808.h"
#include "evmAM1808.h"
#include "hw_types.h"
```

Include dependency graph for syscfg.c:




## Functions

- void SysCfgRegistersUnlock (void)

  *This function unlocks the write-protection of the SYSCFG module.*
- void SysCfgRegistersLock (void)

  *This function imposes write-protection on the SYSCFG module.*

### 4.17.1 Detailed Description

This file contains APIs to lock and unlock the System Configuration (SYSCFG) module registers by appropriately programming the Kick Registers.

### 4.17.2 Function Documentation

#### 4.17.2.1 void SysCfgRegistersLock ( void )

This function imposes write-protection on the SYSCFG module.

**Returns**

None.

**Note**

On programming the Kick Registers with any value other than the unlock sequence, the SYSCFG module gets locked and its registers cannot be accessed.

#### 4.17.2.2 void SysCfgRegistersUnlock ( void )

This function unlocks the write-protection of the SYSCFG module.

**Returns**

None.

**Note**

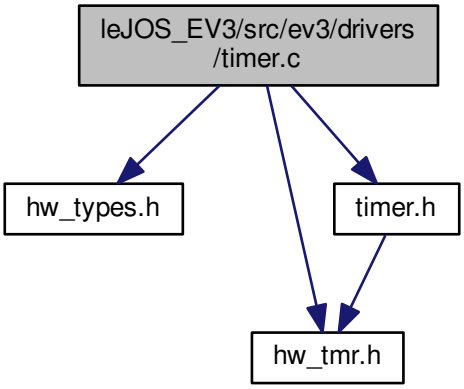
The other registers of the SYSCFG module can be programmed only when an unlock sequence has been written to the Kick Regsiters.

## 4.18 leJOS_EV3/src/ev3/drivers/timer.c File Reference

TIMER APIs.

```
#include "hw_types.h"
#include "hw_tmr.h"
#include "timer.h"
```

Include dependency graph for timer.c:




### Macros

- #define **WDT_KEY_PRE_ACTIVE** (0xA5C6u)
- #define **WDT_KEY_ACTIVE** (0xDA7Eu)
- #define **CMP_IDX_MASK** (0x07)
- #define **PRESCALE_MASK** (0x0F)
- #define **TDDR_MASK** (0x0F)

### Functions

- void TimerEnable (unsigned int baseAddr, unsigned int timer, unsigned int enaMode)

  *Enables the timer in the specified mode. The timer must be configured before it is enabled. The timer starts running when this API is called.*
- void TimerDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the timer. The timer stops running when this API is called.*
- void TimerConfigure (unsigned int baseAddr, unsigned int config)

  *Configures the timer. The timer can be configured in 64 bit mode 32 bit chained/unchained mode, or as a watchdog timer. The timer can be given external clock input or internal clock input. When this API is called,*
  *> The Timer counters are cleared*
  *> Both the timers are disabled from Reset. Hence, both the timers will start counting when enabled.*
  *.*
- void TimerWatchdogActivate (unsigned int baseAddr)

  *Activate the Watchdog timer. The timer shall be configured as watchdog timer before this API is called. This API writes two keys into the WDTCR in the order to activate the WDT. The user shall call TimerWatchdogReactivate API before the WDT expires, to avoid a reset.*
- void TimerWatchdogReactivate (unsigned int baseAddr)

*Re-activate the Watchdog timer. The WDT shall be enabled before this API is called. The user shall call this API before the WDT expires, to avoid a reset.*

- void TimerPeriodSet (unsigned int baseAddr, unsigned int timer, unsigned int period)

  *Set the Period register(s) of the specified timer(s).*
- unsigned int TimerPeriodGet (unsigned int baseAddr, unsigned int timer)

  *Returns the Period register contents of the specified timer.*
- void TimerCounterSet (unsigned int baseAddr, unsigned int timer, unsigned int counter)

  *Set the Counter register of the specified timer.*
- unsigned int TimerCounterGet (unsigned int baseAddr, unsigned int timer)

  *Returns the Counter register contents of the specified timer.*
- void TimerReloadSet (unsigned int baseAddr, unsigned int timer, unsigned int reload)

  *Set the Reload period of the specified timer.*
- unsigned int TimerReloadGet (unsigned int baseAddr, unsigned int timer)

  *Returns the Reload Period of the specified timer.*
- unsigned int TimerCaptureGet (unsigned int baseAddr, unsigned int timer)

  *Returns the capture value of the specified timer.*
- void TimerCompareSet (unsigned int baseAddr, unsigned int regIndex, unsigned int compare)

  *Set the Compare value of the specified CMP register.*
- unsigned int TimerCompareGet (unsigned int baseAddr, unsigned int regIndex)

  *Returns the Compare value of the specified CMP register.*
- void TimerIntEnable (unsigned int baseAddr, unsigned int intFlags)

  *Enables the specified timer interrupts. The timer interrupts which are to be enabled can be passed as parameter to this function.*
- void TimerIntDisable (unsigned int baseAddr, unsigned int intFlags)

  *Disables the specified timer interrupts.*
- unsigned int TimerIntStatusGet (unsigned int baseAddr, unsigned int statFlag)

  *Returns the status of specified timer interrupts.*
- unsigned int TimerIntStatusClear (unsigned int baseAddr, unsigned int statFlag)

  *Clears the status of specified timer interrupts.*
- void TimerPreScalarCount34Set (unsigned int baseAddr, unsigned int psc34)

  *Sets the Prescalar Counter of Timer34.*
- unsigned int TimerPreScalarCount34Get (unsigned int baseAddr)

  *Returns the Prescalar Counter of Timer34.*
- void TimerDivDwnRatio34Set (unsigned int baseAddr, unsigned int tddr34)

  *Sets the Timer Divide Down Ratio of Timer34.*
- unsigned int TimerDivDwnRatio34Get (unsigned int baseAddr)

  *returns the Timer Divide Down Ratio of Timer34.*
- void TimerCaptureConfigure (unsigned int baseAddr, unsigned int timer, unsigned int cfgCap)

  *Configures the Timer for Capture Mode. The Timer Module Shall be Configured in 32 bit unchained mode before this API is called.*
- void TimerReadResetEnable (unsigned int baseAddr, unsigned int timer)

  *Enables the timer(s) for read reset mode. The timer shall be Configured in 32 bit unchained mode before this API is called. Read reset determines the effect of timer counter read on TIMn. If Read reset is enabled, the timer counter will be reset when the timer counter register TIMn is read.*
- void TimerReadResetDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the timer for read reset mode.*
- void TimerInputGateEnable (unsigned int baseAddr, unsigned int timer)

  *Sets the Input Gate Enable. Allows the timer to gate the internal timer clock source. The timer starts counting when the input pin goes from Low to High and stops counting when transition happens from high to low.*
- void TimerInputGateDisable (unsigned int baseAddr, unsigned int timer)

  *Disable the Input Gate. Timer clock will not be gated by input pin.*
- void TimerPulseWidthSet (unsigned int baseAddr, unsigned int timer, unsigned int pulseWidth)

*Sets the pulse width for the specified timer. Determines the pulse. width in the TSTATn bit and the OUT pin in pulse mode.*

- void TimerClockModeSet (unsigned int baseAddr, unsigned int timer)

  *Sets the clock mode. Once the clock mode is set, the outpin behaves as 50% duty cycle signal.*
- void TimerPulseModeSet (unsigned int baseAddr, unsigned int timer)

  *Sets the pulse mode. The outpin goes active when the timer count reaches the period.*
- unsigned int TimerOUTStatusGet (unsigned int baseAddr, unsigned int timer)

  *Returns the timer status. The timer status Drives the value of the timer output TM64P_OUTn when configured.*
- void TimerInvertINEnable (unsigned int baseAddr, unsigned int timer)

  *Inverts the TMR64P_INn signal.*
- void TimerInvertINDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the Inversion of the TMR64P_INn signal.*
- void TimerInvertOUTEnable (unsigned int baseAddr, unsigned int timer)

  *Inverts the TMR64P_OUTn signal.*
- void TimerInvertOUTDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the inversion the TMR64P_OUTn signal.*

### 4.18.1 Detailed Description

TIMER APIs.

This file contains the device abstraction layer APIs for Timer Plus.

### 4.18.2 Function Documentation

#### 4.18.2.1 void TimerCaptureConfigure ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *cfgCap* )

Configures the Timer for Capture Mode. The Timer Module Shall be Configured in 32 bit unchained mode before this API is called.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which capture to be configured. |
| *cfgCap* | Configuration of Capture Mode. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers
cfgCap can take the values
TMR_CAPT_DISABLE - Capture Mode disable
TMR_CAPT_ENABLE_RIS_EDGE - Capture enable at rising edge
TMR_CAPT_ENABLE_FALL_EDGE - Capture enable at falling edge
TMR_CAPT_ENABLE_BOTH_EDGE - Capture enable at both edges

**Returns**

None.

#### 4.18.2.2 unsigned int TimerCaptureGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the capture value of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which capture value to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Capture Value

#### 4.18.2.3 void TimerClockModeSet ( unsigned int *baseAddr,* unsigned int *timer* )

Sets the clock mode. Once the clock mode is set, the outpin behaves as 50% duty cycle signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which clock mode to be set. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.4 unsigned int TimerCompareGet ( unsigned int *baseAddr,* unsigned int *regIndex* )

Returns the Compare value of the specified CMP register.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *regIndex* | Index of the CMP Register |

regIndex can take any value from 0 to 7. If regIndex = n, contents of CMPn will be returned.

**Returns**

Compare Value.

#### 4.18.2.5 void TimerCompareSet ( unsigned int *baseAddr,* unsigned int *regIndex,* unsigned int *compare* )

Set the Compare value of the specified CMP register.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *regIndex* | Index of the CMP Register |
| *compare* | The value to be written |

regIndex can take any value from 0 to 7. If regIndex = n, CMPn will be set with the value compare.

**Returns**

None.

#### 4.18.2.6 void TimerConfigure ( unsigned int *baseAddr,* unsigned int *config* )

Configures the timer. The timer can be configured in 64 bit mode 32 bit chained/unchained mode, or as a watchdog timer. The timer can be given external clock input or internal clock input. When this API is called,
> The Timer counters are cleared
> Both the timers are disabled from Reset. Hence, both the timers will start counting when enabled.
.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *config* | Configuration of the Timer Module. |

config can take the values
TMR_CFG_64BIT_CLK_INT - 64 bit mode with internal clock
TMR_CFG_64BIT_CLK_EXT - 64 bit mode with external clock
TMR_CFG_64BIT_WATCHDOG - 64 bit watchdog timer mode
TMR_CFG_32BIT_CH_CLK_INT - 32 bit chained mode with internal clock
TMR_CFG_32BIT_CH_CLK_EXT - 32 bit chained mode with external clock
TMR_CFG_32BIT_UNCH_CLK_BOTH_INT - 32 bit unchained mode; Both timers clock sources are internal
TMR_CFG_32BIT_UNCH_CLK_12INT_34EXT - 32 bit unchained mode; Clock source for Timer12 is internal and for Timer34 is external
TMR_CFG_32BIT_UNCH_CLK_12EXT_34INT - 32 bit unchained mode; Clock source for Timer12 is external and for Timer34 is internal
TMR_CFG_32BIT_UNCH_CLK_BOTH_EXT - 32 bit unchained mode; Both timers clock sources are external

**Returns**

None.

#### 4.18.2.7 unsigned int TimerCounterGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the Counter register contents of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which counter to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Counter Value.

#### 4.18.2.8 void TimerCounterSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *counter* )

Set the Counter register of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which counter to be set. |
| *counter* | The counter value of the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.9 void TimerDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the timer. The timer stops running when this API is called.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | Timer to be disabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.10 unsigned int TimerDivDwnRatio34Get ( unsigned int *baseAddr* )

returns the Timer Divide Down Ratio of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

TDDR34 value.

#### 4.18.2.11 void TimerDivDwnRatio34Set ( unsigned int *baseAddr,* unsigned int *tddr34* )

Sets the Timer Divide Down Ratio of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *tddr34* | TDDR34, Timer Divide Down Ratio |

**Returns**

None.

#### 4.18.2.12 void TimerEnable ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *enaMode* )

Enables the timer in the specified mode. The timer must be configured before it is enabled. The timer starts running when this API is called.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer to be enabled. |
| *enaMode* | Mode of enabling the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer 12 only
TMR_TIMER_BOTH - Both timers
enaMode can take the values
TMR_ENABLE_ONCE - Enable the timer to run once
TMR_ENABLR_CONT - Enable to run continuous
TMR_ENABLE_CONTRELOAD - Enable to run continuous with period reload

**Returns**

None.

#### 4.18.2.13 void TimerInputGateDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disable the Input Gate. Timer clock will not be gated by input pin.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which input gate to be disabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.14 void TimerInputGateEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Sets the Input Gate Enable. Allows the timer to gate the internal timer clock source. The timer starts counting when the input pin goes from Low to High and stops counting when transition happens from high to low.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which input gate to be enabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.15 void TimerIntDisable ( unsigned int *baseAddr,* unsigned int *intFlags* )

Disables the specified timer interrupts.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *intFlags* | Timer Interrupts to be disabled |

intFlags can take any, or a combination of the following values
TMR_INT_TMR12_CAPT_MODE - Disable Timer12 interrupt in Capture Mode
TMR_INT_TMR12_NON_CAPT_MODE - Disable Timer12 interrupt in normal mode
TMR_INT_TMR34_CAPT_MODE - Disable Timer34 interrupt in Capture mode
TMR_INT_TMR34_NON_CAPT_MODE - Disable Timer34 interrupt in normal mode

**Returns**

None.

#### 4.18.2.16 void TimerIntEnable ( unsigned int *baseAddr,* unsigned int *intFlags* )

Enables the specified timer interrupts. The timer interrupts which are to be enabled can be passed as parameter to this function.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *intFlags* | Timer Interrupts to be enabled |

intFlags can take any, or a combination of the following values
TMR_INT_TMR12_CAPT_MODE - Enable Timer12 interrupt in Capture Mode
TMR_INT_TMR12_NON_CAPT_MODE - Enable Timer12 interrupt in normal mode
TMR_INT_TMR34_CAPT_MODE - Enable Timer34 interrupt in Capture mode
TMR_INT_TMR34_NON_CAPT_MODE - Enable Timer34 interrupt in normal mode

**Returns**

None.

#### 4.18.2.17 unsigned int TimerIntStatusClear ( unsigned int *baseAddr,* unsigned int *statFlag* )

Clears the status of specified timer interrupts.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *statFlag* | Status flags to be cleared. |

intFlags can take any or combination of the following values
TMR_INTSTAT12_TIMER_NON_CAPT - Timer12 interrupt status in normal mode
TMR_INTSTAT12_TIMER_CAPT - Timer12 interrupt status in capture mode
TMR_INTSTAT34_TIMER_NON_CAPT - Timer34 interrupt status in normal mode
TMR_INTSTAT34_TIMER_CAPT - Timer34 interrupt status in capture mode

**Returns**

None

#### 4.18.2.18 unsigned int TimerIntStatusGet ( unsigned int *baseAddr,* unsigned int *statFlag* )

Returns the status of specified timer interrupts.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *statFlag* | Status flags to be read. |

intFlags can take any or combination of the following values
TMR_INTSTAT12_TIMER_NON_CAPT - Timer12 interrupt status in normal mode
TMR_INTSTAT12_TIMER_CAPT - Timer12 interrupt status in capture mode
TMR_INTSTAT34_TIMER_NON_CAPT - Timer34 interrupt status in normal mode
TMR_INTSTAT34_TIMER_CAPT - Timer34 interrupt status in capture mode

**Returns**

Status of Interrupt. Returns all the fields of which status is set

Note : This API will return the same fields which is passed as parameter, if all the specified interrupt status is set. The return value will be 0 if none of the interrupt status in the parameter passed is set.

#### 4.18.2.19 void TimerInvertINDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the Inversion of the TMR64P_INn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be disabled. |

timer can take the values
TMR_TIMER34 - TMR64P_INT34 inversion will be disabled
TMR_TIMER12 - TMR64P_IN12 inversion will be disabled
TMR_TIMER_BOTH - Both TMR64P_IN12 and TMR64P_IN34 inversion disabled

**Returns**

None.

#### 4.18.2.20 void TimerInvertINEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Inverts the TMR64P_INn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be enabled. |

timer can take the following values.
TMR_TIMER34 - Inverts TMR64P_IN34 signal
TMR_TIMER12 - Inverts TMR64P_IN12 signal
TMR_TIMER_BOTH - Inverts both TMR64P_IN12 and TMR64P_IN34 signals

**Returns**

None.

#### 4.18.2.21 void TimerInvertOUTDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the inversion the TMR64P_OUTn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be disabled. |

timer can take the values
TMR_TIMER34 - TMR64P_OUT34 inversion will be disabled
TMR_TIMER12 - TMR64P_OUT12 inversion will be disabled
TMR_TIMER_BOTH - Both TMR64P_OUT12 and TMR64P_OUT34 inversion disabled

**Returns**

None.

#### 4.18.2.22 void TimerInvertOUTEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Inverts the TMR64P_OUTn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be enabled. |

timer can take the values
TMR_TIMER34 - Inverts TMR64P_OUT34 signal
TMR_TIMER12 - Inverts TMR64P_OUT12 signal
TMR_TIMER_BOTH - Inverts both TMR64P_OUT12 and TMR64P_OUT34 signals

**Returns**

None.

#### 4.18.2.23 unsigned int TimerOUTStatusGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the timer status. The timer status Drives the value of the timer output TM64P_OUTn when configured.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which status to be read |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12
TMR_TIMER_BOTH - Both timers
Note : This API returns 0 if none of the status bits is set.

**Returns**

Status of the timer. Returns the following values or the combination of both.
TMR_OUT12_ASSERTED - TMR64P_OUT12 is asserted
TMR_OUT34_ASSERTED - TMR64P_OUT34 is asserted

#### 4.18.2.24 unsigned int TimerPeriodGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the Period register contents of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which period to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Period Value

#### 4.18.2.25 void TimerPeriodSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *period* )

Set the Period register(s) of the specified timer(s).

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which period to be set. |
| *period* | The period value of the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.26 unsigned int TimerPreScalarCount34Get ( unsigned int *baseAddr* )

Returns the Prescalar Counter of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

Prescalar Value.

#### 4.18.2.27 void TimerPreScalarCount34Set ( unsigned int *baseAddr,* unsigned int *psc34* )

Sets the Prescalar Counter of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *psc34* | 4 bit prescalar value |

**Returns**

None.

#### 4.18.2.28 void TimerPulseModeSet ( unsigned int *baseAddr,* unsigned int *timer* )

Sets the pulse mode. The outpin goes active when the timer count reaches the period.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which pulse mode to be set. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.29 void TimerPulseWidthSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *pulseWidth* )

Sets the pulse width for the specified timer. Determines the pulse. width in the TSTATn bit and the OUT pin in pulse mode.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which pulse width to be set. |
| *pulseWidth* | Pulse width to be set. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers
pulseWidth can take the following values
TMR_PULSE_WIDTH_1_CLK - 1 clock cycle
TMR_PULSE_WIDTH_2_CLK - 2 clock cycles
TMR_PULSE_WIDTH_3_CLK - 3 clock cycles
TMR_PULSE_WIDTH_4_CLK - 4 clock cycles

**Returns**

None.

#### 4.18.2.30 void TimerReadResetDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the timer for read reset mode.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which Read Reset to be disabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.31 void TimerReadResetEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Enables the timer(s) for read reset mode. The timer shall be Configured in 32 bit unchained mode before this API is called. Read reset determines the effect of timer counter read on TIMn. If Read reset is enabled, the timer counter will be reset when the timer counter register TIMn is read.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which Read Reset to be enabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.32 unsigned int TimerReloadGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the Reload Period of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which Reload value to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Reload Value

#### 4.18.2.33 void TimerReloadSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *reload* )

Set the Reload period of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which reload period to be set. |
| *reload* | The reload period value of the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.18.2.34 void TimerWatchdogActivate ( unsigned int *baseAddr* )

Activate the Watchdog timer. The timer shall be configured as watchdog timer before this API is called. This API writes two keys into the WDTCR in the order to activate the WDT. The user shall call TimerWatchdogReactivate API before the WDT expires, to avoid a reset.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

None.

#### 4.18.2.35 void TimerWatchdogReactivate ( unsigned int *baseAddr* )

Re-activate the Watchdog timer. The WDT shall be enabled before this API is called. The user shall call this API before the WDT expires, to avoid a reset.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

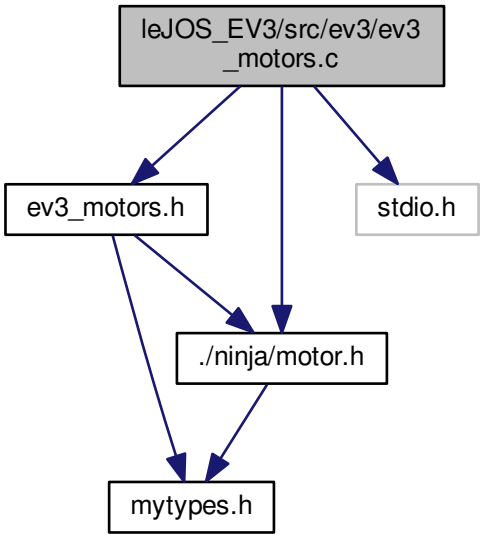
None.

## 4.19 leJOS_EV3/src/ev3/ev3_motors.c File Reference

This file contains function definitions to use EV3 motors. Driver wrapper for ECRobot API.

```
#include "ev3_motors.h"
#include "ninja/motor.h"
#include "stdio.h"
```

Include dependency graph for ev3_motors.c:




## Functions

- int nxt_motor_get_count (U32 n)

  *Get motor revolution count in degree.*
- void nxt_motor_set_count (U32 n, int count)

  *Set motor revolution count in degree.*
- void nxt_motor_set_speed (U32 n, int speed_percent, int brake)

  *Set motor speed and brake mode.*
- void nxt_motor_command (U32 n, int cmd, int target_count, int speed_percent)

  *Set motor target revolution count to reach, speed percent and brake mode.*

### 4.19.1 Detailed Description

This file contains function definitions to use EV3 motors. Driver wrapper for ECRobot API.

**Author**

### 4.19.2 Function Documentation

#### 4.19.2.1 void nxt_motor_command ( U32 *n,* int *cmd,* int *target_count,* int *speed_percent* )

Set motor target revolution count to reach, speed percent and brake mode.

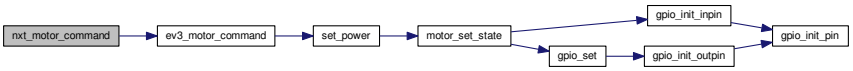
**Parameters**

| | |
|---|---|
| *n* | - Motor port id |
| *cmd* | - Brake mode. True - brake, false - stop |
| *target_count* | - Target revolution count to reach and stop |
| *speed_percent* | - Speed percent to rotate |

**Returns**

none

Here is the call graph for this function:




#### 4.19.2.2 int nxt_motor_get_count ( U32 *n* )

Get motor revolution count in degree.

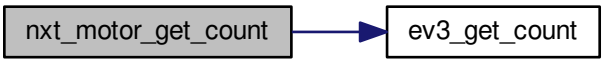
**Parameters**

| | |
|---|---|
| *n* | - Motor port id |

**Returns**

revolution in degree

Here is the call graph for this function:




#### 4.19.2.3 void nxt_motor_set_count ( U32 *n*, int *count* )

Set motor revolution count in degree.

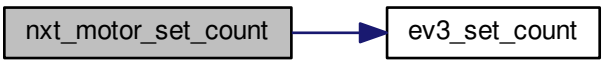
**Parameters**

| | |
|---|---|
| *n* | - Motor port id |
| *count* | - Motor revolution count to set |

**Returns**

none

Here is the call graph for this function:




#### 4.19.2.4 void nxt_motor_set_speed ( U32 *n*, int *speed_percent*, int *brake* )

Set motor speed and brake mode.

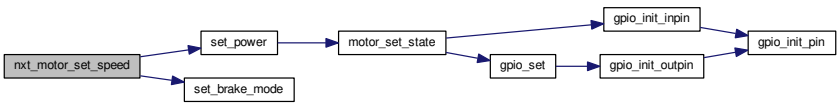
**Parameters**

| | |
|---|---|
| *n* | - Motor port id |
| *speed_percent* | - Speed percent. The given motor rotates backward if negative |
| *brake* | - Brake mode. True - brake (stop immediately), false - float (soft stop) |

**Returns**

none

Here is the call graph for this function:

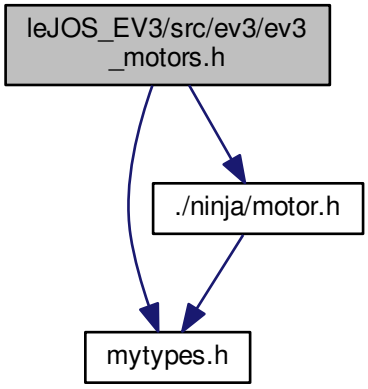



## 4.20 leJOS_EV3/src/ev3/ev3_motors.h File Reference

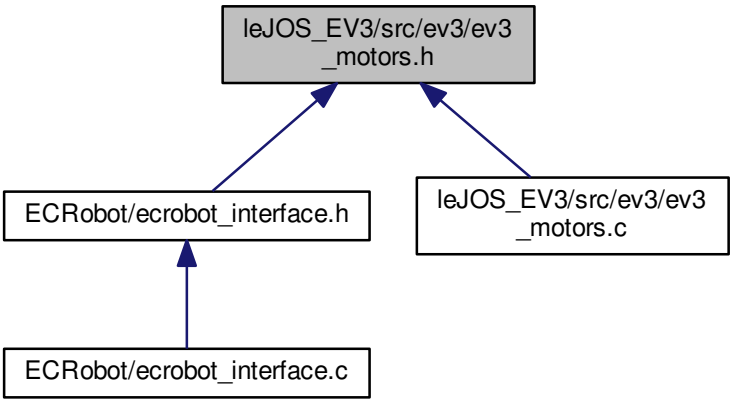
This header contains function declarations to use motor drivers. Driver wrapper for ECRobot API.

```
#include "mytypes.h"
#include "./ninja/motor.h"
```

Include dependency graph for ev3_motors.h:




This graph shows which files directly or indirectly include this file:




## Macros

- #define **EV3_N_MOTORS** 3
- #define **NXT_PORT_A** MOTOR_PORT_A
- #define **NXT_PORT_B** MOTOR_PORT_B
- #define **NXT_PORT_C** MOTOR_PORT_C
- #define **EV3_PORT_A** MOTOR_PORT_A
- #define **EV3_PORT_B** MOTOR_PORT_B
- #define **EV3_PORT_C** MOTOR_PORT_C
- #define **EV3_PORT_D** MOTOR_PORT_D

## Functions

- int nxt_motor_get_count (U32 n)

  *Get motor revolution count in degree.*
- void nxt_motor_set_count (U32 n, int count)

  *Set motor revolution count in degree.*
- void nxt_motor_set_speed (U32 n, int speed_percent, int brake)

  *Set motor speed and brake mode.*
- void nxt_motor_command (U32 n, int cmd, int target_count, int speed_percent)

  *Set motor target revolution count to reach, speed percent and brake mode.*

### 4.20.1 Detailed Description

This header contains function declarations to use motor drivers. Driver wrapper for ECRobot API.

**Author**

### 4.20.2 Function Documentation

#### 4.20.2.1 void nxt_motor_command ( U32 *n*, int *cmd*, int *target_count*, int *speed_percent* )

Set motor target revolution count to reach, speed percent and brake mode.

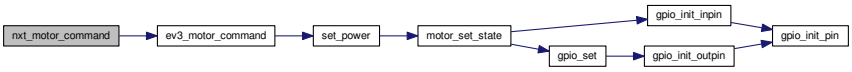
**Parameters**

| | |
|---|---|
| *n* | - Motor port id |
| *cmd* | - Brake mode. True - brake, false - stop |
| *target_count* | - Target revolution count to reach and stop |
| *speed_percent* | - Speed percent to rotate |

**Returns**

none

Here is the call graph for this function:




#### 4.20.2.2 int nxt_motor_get_count ( U32 *n* )

Get motor revolution count in degree.

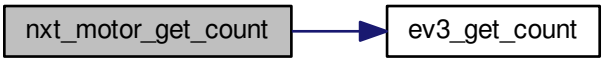
**Parameters**

| | |
|---|---|
| *n* | - Motor port id |

**Returns**

revolution in degree

Here is the call graph for this function:




#### 4.20.2.3 void nxt_motor_set_count ( U32 *n*, int *count* )

Set motor revolution count in degree.

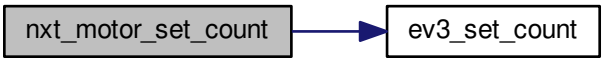
**Parameters**

| | |
|---|---|
| *n* | - Motor port id |
| *count* | - Motor revolution count to set |

**Returns**

none

Here is the call graph for this function:




#### 4.20.2.4 void nxt_motor_set_speed ( U32 *n*, int *speed_percent*, int *brake* )

Set motor speed and brake mode.

**Parameters**

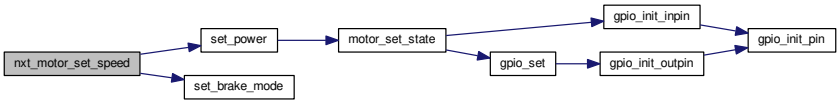
| | |
|---|---|
| *n* | - Motor port id |
| *speed_percent* | - Speed percent. The given motor rotates backward if negative |

| | |
|---|---|
| *brake* | - Brake mode. True - brake (stop immediately), false - float (soft stop) |

**Returns**

none

Here is the call graph for this function:

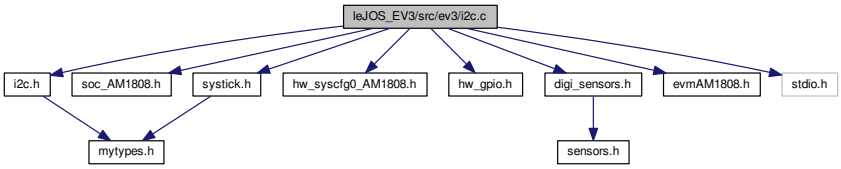



## 4.21 leJOS_EV3/src/ev3/i2c.c File Reference

Function definitions related to I2C communication.

```
#include "i2c.h"
#include "soc_AM1808.h"
#include "systick.h"
#include "hw_syscfg0_AM1808.h"
#include "hw_gpio.h"
#include "digi_sensors.h"
#include "evmAM1808.h"
#include <stdio.h>
```

Include dependency graph for i2c.c:




### Macros

- #define I2C_N_PORTS 4

  *The amount of I2C ports on the EV3.*
- #define I2C_N_RETRIES 3

  *The amount of maximum retries before an I2C transaction fails.*
- #define PINMUX0Register *(unsigned int *)(SOC_SYSCFG_0_REGS + SYSCFG0_PINMUX(0))

  *Pin-multiplexing register 0.*
- #define PINMUX1Register *(unsigned int *)(SOC_SYSCFG_0_REGS + SYSCFG0_PINMUX(1))

  *Pin-multiplexing register 1.*
- #define PINMUX2Register *(unsigned int *)(SOC_SYSCFG_0_REGS + SYSCFG0_PINMUX(2))

  *Pin-multiplexing register 2.*
- #define MASK_PORT_1_PINMUX0_CLEAR 0xFFFFFFF0

  *Bitmask to clear the half-byte representing the pin-multiplexing configuration for Port 1 (SDA)*

- #define MASK_PORT_1_PINMUX0_SET 0x00000008

  *Bitmask to set the half-byte representing the pin-multiplexing configuration for Port 1 (SDA)*
- #define MASK_PORT_1_PINMUX1_CLEAR 0xFF0FFFFF

  *Bitmask to clear the half-byte representing the pin-multiplexing configuration for Port 1 (SCL)*
- #define MASK_PORT_1_PINMUX1_SET 0x00800000

  *Bitmask to set the half-byte representing the pin-multiplexing configuration for Port 1 (SCL)*
- #define MASK_PORT_2_PINMUX0_CLEAR 0xFFFFF00F

  *Bitmask to clear both half-bytes representing the pin-multiplexing configuration for Port 2 (SDA and SCL)*
- #define MASK_PORT_2_PINMUX0_SET 0x00000880

  *Bitmask to set both half-bytes representing the pin-multiplexing configuration for Port 2 (SDA and SCL)*
- #define MASK_PORT_3_PINMUX0_CLEAR 0xFFFF0FFF

  *Bitmask to clear the half-byte representing the pin-multiplexing configuration for Port 3 (SDA)*
- #define MASK_PORT_3_PINMUX0_SET 0x00008000

  *Bitmask to set the half-byte representing the pin-multiplexing configuration for Port 3 (SDA)*
- #define MASK_PORT_3_PINMUX2_CLEAR 0xFFFFFF0F

  *Bitmask to clear the half-byte representing the pin-multiplexing configuration for Port 3 (SCL)*
- #define MASK_PORT_3_PINMUX2_SET 0x00000040

  *Bitmask to set the half-byte representing the pin-multiplexing configuration for Port 3 (SCL)*
- #define MASK_PORT_4_PINMUX1_CLEAR 0xF0FFFFFF

  *Bitmask to clear the half-byte representing the pin-multiplexing configuration for Port 4 (SDA)*
- #define MASK_PORT_4_PINMUX1_SET 0x08000000

  *Bitmask to set the half-byte representing the pin-multiplexing configuration for Port 4 (SDA)*
- #define MASK_PORT_4_PINMUX2_CLEAR 0xFFFFFF0F

  *Bitmask to clear the half-byte representing the pin-multiplexing configuration for Port 4 (SCL)*
- #define MASK_PORT_4_PINMUX2_SET 0x00000008

  *Bitmask to set the half-byte representing the pin-multiplexing configuration for Port 4 (SCL)*
- #define SetSDADirectionOutput(port) *((unsigned int *)(SOC_GPIO_0_REGS + GPIO_DIR(port.SDA.gpio↩_registers))) &= ~port.SDA_SET_CLEAR

  *Set the SDA pin of the specified port as an output pin.*
- #define SetSCLDirectionOutput(port) *((unsigned int *)(SOC_GPIO_0_REGS + GPIO_DIR(port.SCL.gpio↩_registers))) &= ~port.SCL_SET_CLEAR

  *Set the SCL pin of the specified port as an output pin.*
- #define SetDirectionOutput(port) (SetSDADirectionOutput(port)); (SetSCLDirectionOutput(port));

  *Set the SDA and SCL pins of the specified port as output pins.*
- #define SetSDADirectionInput(port) *((unsigned int *)(SOC_GPIO_0_REGS + GPIO_DIR(port.SDA.gpio_↩registers))) |= port.SDA_SET_CLEAR

  *Set the SDA pin of the specified port as an input pin.*
- #define SetSCLDirectionInput(port) *((unsigned int *)(SOC_GPIO_0_REGS + GPIO_DIR(port.SCL.gpio_↩registers))) |= port.SCL_SET_CLEAR

  *Set the SCL pin of the specified port as an input pin.*
- #define SetDirectionInput (SetSDADirectionInput(port)); (SetSCLDirectionInput(port));

  *Set the SDA and SCL pins of the specified port as input pins.*
- #define SDASetDataRegister(port) (unsigned int *)(SOC_GPIO_0_REGS + GPIO_SET_DATA(port.SDA.↩gpio_registers))

  *Pointer to register required to set the SDA pin of the specified port high.*
- #define SCLSetDataRegister(port) (unsigned int *)(SOC_GPIO_0_REGS + GPIO_SET_DATA(port.SCL.↩gpio_registers))

  *Pointer to register required to set the SCL pin of the specified port high.*
- #define SDAClearDataRegister(port) (unsigned int *)(SOC_GPIO_0_REGS + GPIO_CLR_DATA(port.SD↩A.gpio_registers))

  *Pointer to register required to set the SCL pin of the specified port low.*

- #define SCLClearDataRegister(port) (unsigned int *)(SOC_GPIO_0_REGS + GPIO_CLR_DATA(port.SCL.gpio_registers))

  *Pointer to register required to set the SCL pin of the specified port low.*
- #define SDAInRegister(port) (unsigned int *)(SOC_GPIO_0_REGS + GPIO_IN_DATA(port.SDA.gpio_registers))

  *Pointer to register required to read the SDA pin of the specified port.*
- #define SCLInRegister(port) (unsigned int *)(SOC_GPIO_0_REGS + GPIO_IN_DATA(port.SCL.gpio_registers))

  *Pointer to register required to read the SCL pin of the specified port.*
- #define SDA_HIGH(port) (*(SDASetDataRegister(port)) |= port.SDA_SET_CLEAR);

  *Set the SDA pin of the specified port high (needs to be configured as output)*
- #define SDA_LOW(port) (*(SDAClearDataRegister(port)) &= port.SDA_SET_CLEAR);

  *Set the SDA pin of the specified port low (needs to be configured as output)*
- #define SCL_HIGH(port) (*(SCLSetDataRegister(port)) |= port.SCL_SET_CLEAR);

  *Set the SCL pin of the specified port high (needs to be configured as output)*
- #define SCL_LOW(port) (*(SCLClearDataRegister(port)) &= port.SCL_SET_CLEAR);

  *Set the SCL pin of the specified port low (needs to be configured as output)*
- #define READ_SDA(port) ((*SDAInRegister(port) & port.SDA_SET_CLEAR) > 0 ? 1 : 0)

  *Read the SDA pin of the specified port (needs to be configured as input)*
- #define READ_SCL(port) ((*SCLInRegister(port) & port.SCL_SET_CLEAR) > 0 ? 1 : 0)

  *Read the SCL pin of the specified port (needs to be configured as input)*
- #define DELAY doDelay(85);

  *Delay the communication between two clock pulses.*

## Functions

- void doDelay (int time)

  *Delay the communication between two clock pulses by counting from 0 to the specified value.*
- void i2c_write_bit (I2C_PORT port, U8 bit)

  *Send a bit by setting the SDA pin high or low.*
- U8 i2c_read_bit (I2C_PORT port)

  *Read a bit by reading the signal on SDA.*
- U8 i2c_write_byte (I2C_PORT port, U8 byte)

  *Send a byte.*
- U8 i2c_read_byte (I2C_PORT port)

  *Read a byte.*
- void i2c_send_start (I2C_PORT port)

  *Send an I2C start signal.*
- U8 i2c_send_slave_address (I2C_PORT port, U8 slave_address, U8 read_write_bit)

  *Send the I2C slave address along with the read/write bit.*
- void i2c_send_stop (I2C_PORT port)

  *Send an I2C stop signal.*
- int i2c_available (int port, U32 address)

  *Check if an I2C device with the given address is available at the specified with port.*
- void i2c_disable (int port)

  *Disable an I2C port.*
- void i2c_enable (int port)

  *Enable an I2C port.*
- void i2c_init (void)

  *Initialize the I2C module.*
- int i2c_busy (int port)

*Check whether an I2C port is currently busy or not.*

- int i2c_start_transaction (int port, U32 address, int internal_address, int n_internal_address_bytes, U8 *data, U32 nbytes, int write)

  *Start a new I2C transaction AND wait until it's finished.*

## Variables

- I2C_PORT ports_i2c [I2C_N_PORTS]

  *This array stores information about the GPIO pins which represent SDA and SCL lines for the I2C communication.*

### 4.21.1 Detailed Description

Function definitions related to I2C communication.

I2C communication is required in order to interact with the digital sensors.

**Author**

Tobias Schießl

### 4.21.2 Macro Definition Documentation

#### 4.21.2.1 #define DELAY doDelay(85);

Delay the communication between two clock pulses.

Counting to 85 is the best value to get as close to the original Lego firmware as possible (meassured with an oscilloscope). With this value, we have a delay between 100 and 125 ns between two clock pulses.

### 4.21.3 Function Documentation

#### 4.21.3.1 void doDelay ( int *time* )

Delay the communication between two clock pulses by counting from 0 to the specified value.

This approach is necessary since we cannot wait below 1 ms with our systick impplementation and this is far too slow for I2C communication.

**Parameters**

| | |
|---|---|
| *time* | - Ther value up to which the counter will be incremented before returning |

**Returns**

none

#### 4.21.3.2 int i2c_available ( int *port*, U32 *address* )

Check if an I2C device with the given address is available at the specified with port.

This function makes use of the internal register layout which is equal among all Lego/HiTechnic I2C sensors. On internal address 0x00, the product version if the sensor is stored. If we read this value and receive a 0, we can assume that no sensor is available since a product version of 0 makes no sense. So far this is the only way to figure out if a sensor is connected to the port. Since the SDA signal will default to 0 when configured as input, the ACK bit is not enough to check that. It would always be 0 and therefore we would assume that a device is ready.

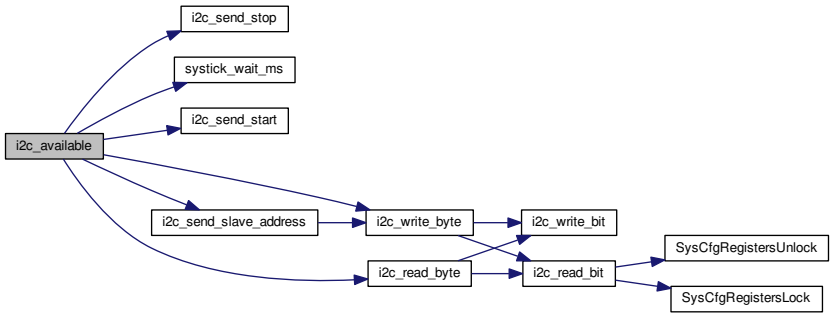
**Parameters**

| | |
|---|---|
| *port* | - The port in use |
| *address* | - The I2C slave address of the slave device |

**Returns**

0 if no device is available, 1 otherwise

Here is the call graph for this function:




#### 4.21.3.3 int i2c_busy ( int *port* )

Check whether an I2C port is currently busy or not.

Since our I2C communication is currently not based on interrupts (as it is in the NXT version of the leJOS driver) a port is never busy. Therefore this function will always return 0.

**Parameters**

| | |
|---|---|
| *port* | - The port to check |

**Returns**

Always 0 since our port can never be busy (no transaction will be pending)

#### 4.21.3.4 void i2c_disable ( int *port* )

Disable an I2C port.

This function currently has no functionality and therefore it is not necessary to call it.

**Parameters**

| | |
|---|---|
| *port* | - The port to disable |

**Returns**

none

#### 4.21.3.5 void i2c_enable ( int *port* )

Enable an I2C port.

Enabling a port means configuring the corresponding pin-multiplexing registers and setting the pins as output pins with a high signal.

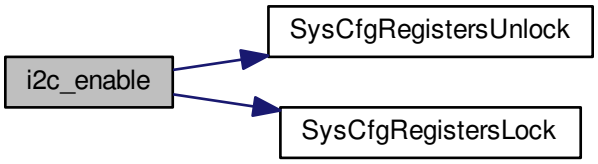
**Parameters**

| | |
|---|---|
| *port* | - The port to disable |

**Returns**

none

Here is the call graph for this function:




#### 4.21.3.6 void i2c_init ( void )

Initialize the I2C module.

This function currently has no functionality and therefore it is not necessary to call it.

**Returns**

none

#### 4.21.3.7 U8 i2c_read_bit ( I2C_PORT *port* )

Read a bit by reading the signal on SDA.

Only the last bit of the returned byte should be considered (LSB). All other bits will be 0. This function will set SDA as an input pin in order to give the control to the slave device (sensor). The pin will be reset to output before returning from this function. SCL should be low before calling this function and will be low again after returning from it.

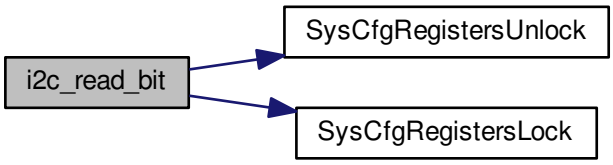
**Parameters**

| | |
|---|---|
| *port* | - The port in use |

**Returns**

The bit red (LSB of the returned byte)

Here is the call graph for this function:




#### 4.21.3.8 U8 i2c_read_byte ( I2C_PORT *port* )

Read a byte.

The byte will be red by reading it bit by bit. Therefore i2c_read_bit will be used. Before returning, this function will send an ACK bit to the slave device (sensor). SCL should be low before calling this function and will be low again after returning from it.

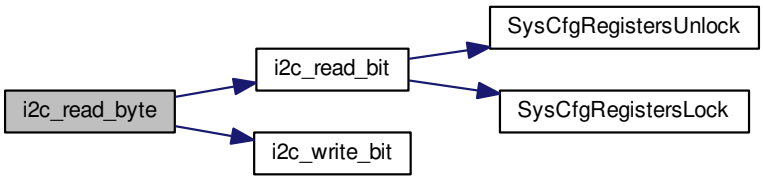
**Parameters**

| | |
|---|---|
| *port* | - The port in use |

**Returns**

The byte red

Here is the call graph for this function:




#### 4.21.3.9 U8 i2c_send_slave_address ( I2C_PORT *port*, U8 *slave_address*, U8 *read_write_bit* )

Send the I2C slave address along with the read/write bit.

This is the first byte that has to be sent in an I2C transaction after sending the start signal. The slave address is a 7 bit address which gets shifted by 1 to the left inside this function. The last bit of the byte will be 1 if we want to write to the slave device (sensor), 0 if we want to read from it. SCL should be low before calling this function and will be low again after returning from it.

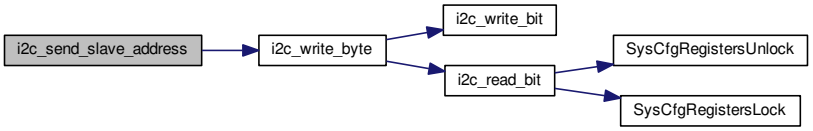
**Parameters**

| | |
|---|---|
| *port* | - The port in use |
| *slave_address* | - The slave address to use (unshifted) |
| *read_write_bit* | - The read/write bit (1 if we want to write to the slave device, 0 if we want to read from it) |

**Returns**

The ACK bit received from the slave (0 for ACK and 1 for no ACK)

Here is the call graph for this function:




#### 4.21.3.10 void i2c_send_start ( I2C_PORT *port* )

Send an I2C start signal.

A start in an I2C transaction means setting SDA to 0 while SCL is 1. After calling this function, SDA and SCL will both be low.

**Parameters**

| | |
|---|---|
| *port* | - The port in use |

**Returns**

none

#### 4.21.3.11 void i2c_send_stop ( I2C_PORT *port* )

Send an I2C stop signal.

A stop in an I2C transaction means setting SDA to 1 while SCL is 1. After calling this function, SDA and SCL will both be high.

**Parameters**

| | |
|---|---|
| *port* | - The port in use |

**Returns**

none

#### 4.21.3.12 int i2c_start_transaction ( int *port*, U32 *address*, int *internal_address*, int *n_internal_address_bytes*, U8 * *data*, U32 *nbytes*, int *write* )

Start a new I2C transaction AND wait until it's finished.

In contrast to the name of the function, this will not only start the I2C transaction. Instead, the whole transaction is processed before returning from this function. The name remains in order to be API-compatible with the NXT version of the leJOS driver. In this driver, the I2C communication was controlled using interrupts. Note: According to the I2C protocol, a write is represented by a 0 and not by a 1 as in this function call. This behaviour is intended an also to keep API-compatibility to the NXT version of the driver.

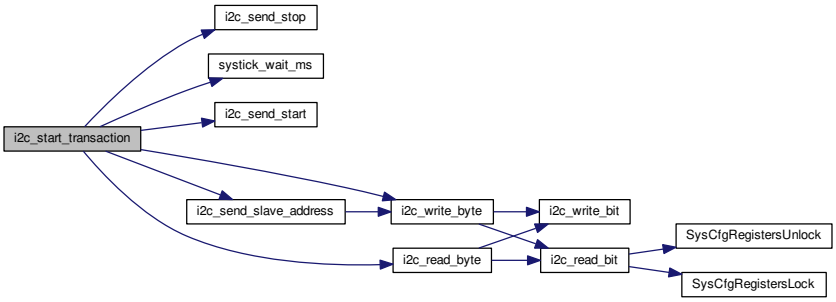
**Parameters**

| | |
|---|---|
| *port* | - The port to use (0 to 3) |
| *address* | - The I2C slave address of the slave device |
| *internal_address* | - Internal register address of slave device to read from or write to |
| *n_internal_address_bytes* | - Size of the internal register address in bytes |
| *data* | - Buffer to store received data in or the data to send |
| *nbytes* | - Amount of bytes to receive or send, i.e. the length of the data array |
| *write* | - 1 if we want to write to the slave, 0 if we want to read from it |

**Returns**

0 if the transaction was successful, otherwise the number of the byte for which no ACK was received (beginning with 1)

Here is the call graph for this function:




#### 4.21.3.13 void i2c_write_bit ( I2C_PORT *port*, U8 *bit* )

Send a bit by setting the SDA pin high or low.

Only the last bit of the given byte will be used (LSB). SCL should be low before calling this function and will be low again after returning from it.

**Parameters**

| | |
|---|---|
| *port* | - The port in use |
| *bit* | - The bit to write (only the LSB of the byte will be considered) |

**Returns**

none

#### 4.21.3.14 U8 i2c_write_byte ( I2C_PORT *port*, U8 *byte* )

Send a byte.

The byte will be written by sending it bit by bit. Therefore i2c_write_bit will be used. SCL should be low before calling this function and will be low again after returning from it.

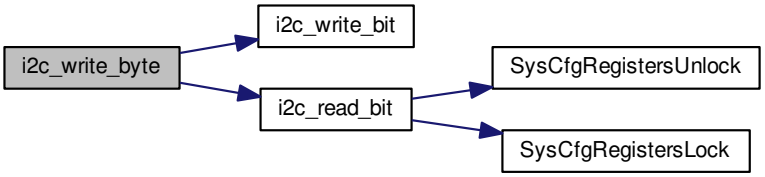
**Parameters**

| | |
|---|---|
| *port* | - The port in use |
| *byte* | - The byte to send |

**Returns**

The ACK bit received from the slave (0 for ACK and 1 for no ACK)

Here is the call graph for this function:




### 4.21.4 Variable Documentation

#### 4.21.4.1 I2C_PORT ports_i2c[I2C_N_PORTS]

**Initial value:**

```
= {
    {{0, 15}, {0, 2}, 0x00008000, 0x00000004},
    {{0, 13}, {0, 14}, 0x00002000, 0x00004000},
    {{0, 14}, {0, 12}, 0x40000000, 0x00001000},
    {{0, 15}, {0, 1}, 0x80000000, 0x00000002}
}
```

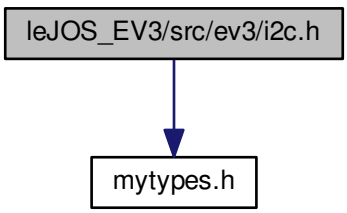
This array stores information about the GPIO pins which represent SDA and SCL lines for the I2C communication.

## 4.22 leJOS_EV3/src/ev3/i2c.h File Reference

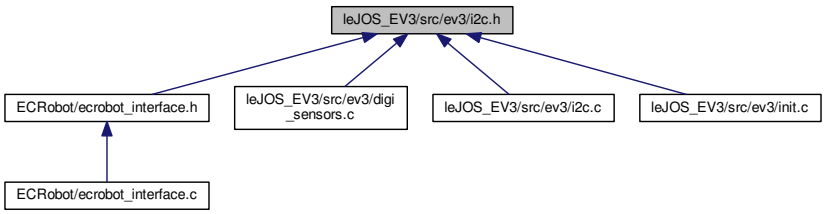
Function declarations related to I2C communication.

```
#include "mytypes.h"
```

Include dependency graph for i2c.h:



This graph shows which files directly or indirectly include this file:




## Data Structures

- struct I2C_PIN

  *This struct represents a GPIO pin used for I2C communication.*
- struct I2C_PORT

  *This struct represents an I2C port.*

## Typedefs

- typedef struct I2C_PIN I2C_PIN

  *This struct represents a GPIO pin used for I2C communication.*
- typedef struct I2C_PORT I2C_PORT

  *This struct represents an I2C port.*

## Functions

- void i2c_disable (int port)

  *Disable an I2C port.*
- void i2c_enable (int port)

  *Enable an I2C port.*
- void i2c_init (void)

  *Initialize the I2C module.*
- int i2c_busy (int port)

  *Check whether an I2C port is currently busy or not.*
- int i2c_start_transaction (int port, U32 address, int internal_address, int n_internal_address_bytes, U8 *data, U32 nbytes, int write)

  *Start a new I2C transaction AND wait until it's finished.*

### 4.22.1 Detailed Description

Function declarations related to I2C communication.

I2C communication is required in order to interact with the digital sensors.

**Author**

Tobias Schießl

### 4.22.2 Typedef Documentation

#### 4.22.2.1 typedef struct I2C_PIN I2C_PIN

This struct represents a GPIO pin used for I2C communication.

Every pin is represented by a pin number on a GPIO bank (0 to 15) and one GPIO register to control that pin. Note: 2 GPIO banks always share one GPIO register, therefore bank 0 and 1 relate to register 0, bank 2 and 3 relate to register 1 and so on.

#### 4.22.2.2 typedef struct I2C_PORT I2C_PORT

This struct represents an I2C port.

Every port is represented by two GPIO pins (SDA for data and SCL for clock). This struct also contains the corresponding bitmasks in order to manipulate said GPIO pins.

### 4.22.3 Function Documentation

#### 4.22.3.1 int i2c_busy ( int *port* )

Check whether an I2C port is currently busy or not.

Since our I2C communication is currently not based on interrupts (as it is in the NXT version of the leJOS driver) a port is never busy. Therefore this function will always return 0.

**Parameters**

| | |
|---|---|
| *port* | - The port to check |

**Returns**

Always 0 since our port can never be busy (no transaction will be pending)

#### 4.22.3.2 void i2c_disable ( int *port* )

Disable an I2C port.

This function currently has no functionality and therefore it is not necessary to call it.

**Parameters**

| | |
|---|---|
| *port* | - The port to disable |

**Returns**

none

#### 4.22.3.3 void i2c_enable ( int *port* )

Enable an I2C port.

Enabling a port means configuring the corresponding pin-multiplexing registers and setting the pins as output pins with a high signal.

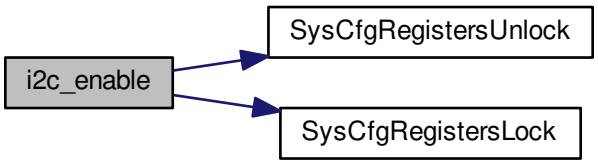
**Parameters**

| | |
|---|---|
| *port* | - The port to disable |

**Returns**

none

Here is the call graph for this function:




#### 4.22.3.4 void i2c_init ( void )

Initialize the I2C module.

This function currently has no functionality and therefore it is not necessary to call it.

**Returns**

none

#### 4.22.3.5 int i2c_start_transaction ( int *port*, U32 *address*, int *internal_address*, int *n_internal_address_bytes*, U8 * *data*, U32 *nbytes*, int *write* )

Start a new I2C transaction AND wait until it's finished.

In contrast to the name of the function, this will not only start the I2C transaction. Instead, the whole transaction is processed before returning from this function. The name remains in order to be API-compatible with the NXT version of the leJOS driver. In this driver, the I2C communication was controlled using interrupts. Note: According to the I2C protocol, a write is represented by a 0 and not by a 1 as in this function call. This behaviour is intended an also to keep API-compatibility to the NXT version of the driver.

**Parameters**

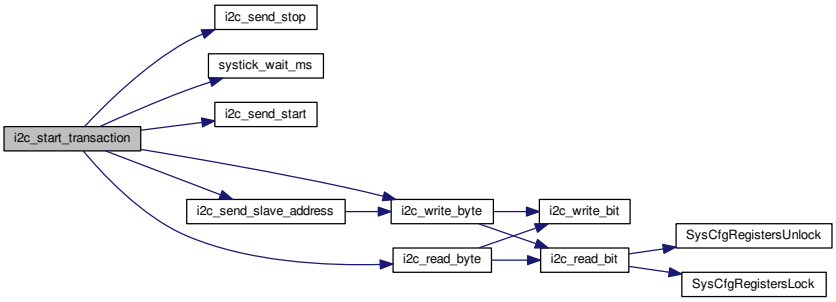
| | |
|---|---|
| *port* | - The port to use (0 to 3) |
| *address* | - The I2C slave address of the slave device |
| *internal_address* | - Internal register address of slave device to read from or write to |
| *n_internal_address_bytes* | - Size of the internal register address in bytes |

| | |
|---|---|
| *data* | - Buffer to store received data in or the data to send |
| *nbytes* | - Amount of bytes to receive or send, i.e. the length of the data array |
| *write* | - 1 if we want to write to the slave, 0 if we want to read from it |

**Returns**

0 if the transaction was successful, otherwise the number of the byte for which no ACK was received (beginning with 1)

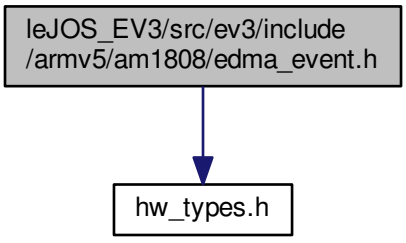
Here is the call graph for this function:




## 4.23 leJOS_EV3/src/ev3/include/armv5/am1808/edma_event.h File Reference

EDMA event enumeration.

`#include "hw_types.h"`
Include dependency graph for edma_event.h:




### Macros

- #define **EDMA3_CHA_MCASP0_RX** 0
- #define **EDMA3_CHA_MCASP0_TX** 1
- #define **EDMA3_CHA_MCBSP0_RX** 2
- #define **EDMA3_CHA_MCBSP0_TX** 3
- #define **EDMA3_CHA_MCBSP1_RX** 4

- #define **EDMA3_CHA_MCBSP1_TX** 5
- #define **EDMA3_CHA_GPIO_BNKINT0** 6
- #define **EDMA3_CHA_GPIO_BNKINT1** 7
- #define **EDMA3_CHA_GPIO_BNKINT2** 22
- #define **EDMA3_CHA_GPIO_BNKINT3** 23
- #define **EDMA3_CHA_GPIO_BNKINT4** 28
- #define **EDMA3_CHA_GPIO_BNKINT5** 29
- #define **EDMA3_CHA_GPIO_BNKINT6** 16
- #define **EDMA3_CHA_GPIO_BNKINT7** 17
- #define **EDMA3_CHA_GPIO_BNKINT8** 18
- #define **EDMA3_CHA_UART0_RX** 8
- #define **EDMA3_CHA_UART0_TX** 9
- #define **EDMA3_CHA_UART1_RX** 12
- #define **EDMA3_CHA_UART1_TX** 13
- #define **EDMA3_CHA_UART2_RX** 30
- #define **EDMA3_CHA_UART2_TX** 31
- #define **EDMA3_CHA_TIMER64P0_EVT12** 10
- #define **EDMA3_CHA_TIMER64P0_EVT34** 11
- #define **EDMA3_CHA_TIMER64P2_EVT12** 24
- #define **EDMA3_CHA_TIMER64P2_EVT34** 25
- #define **EDMA3_CHA_TIMER64P3_EVT12** 26
- #define **EDMA3_CHA_TIMER64P3_EVT34** 27
- #define **EDMA3_CHA_SPI0_RX** 14
- #define **EDMA3_CHA_SPI0_TX** 15
- #define **EDMA3_CHA_SPI1_RX** 18
- #define **EDMA3_CHA_SPI1_TX** 19
- #define **EDMA3_CHA_MMCSD0_RX** 16
- #define **EDMA3_CHA_MMCSD0_TX** 17
- #define **EDMA3_CHA_MMCSD1_RX** 28
- #define **EDMA3_CHA_MMCSD1_TX** 29
- #define **EDMA3_CHA_I2C0_RX** 24
- #define **EDMA3_CHA_I2C0_TX** 25
- #define **EDMA3_CHA_I2C1_RX** 26
- #define **EDMA3_CHA_I2C1_TX** 27
- #define **EDMA3_TIMER2_T12CMPEVT0** 0
- #define **EDMA3_TIMER2_T12CMPEVT1** 1
- #define **EDMA3_TIMER2_T12CMPEVT2** 2
- #define **EDMA3_TIMER2_T12CMPEVT3** 3
- #define **EDMA3_TIMER2_T12CMPEVT4** 4
- #define **EDMA3_TIMER2_T12CMPEVT5** 5
- #define **EDMA3_TIMER2_T12CMPEVT6** 6
- #define **EDMA3_TIMER2_T12CMPEVT7** 7
- #define **EDMA3_TIMER3_T12CMPEVT0** 8
- #define **EDMA3_TIMER3_T12CMPEVT1** 9
- #define **EDMA3_TIMER3_T12CMPEVT2** 10
- #define **EDMA3_TIMER3_T12CMPEVT3** 11
- #define **EDMA3_TIMER3_T12CMPEVT4** 12
- #define **EDMA3_TIMER3_T12CMPEVT5** 13
- #define **EDMA3_TIMER3_T12CMPEVT6** 14
- #define **EDMA3_TIMER3_T12CMPEVT7** 15
- #define **EDMA3_PRU_EVTOUT6** 20
- #define **EDMA3_PRU_EVTOUT7** 21

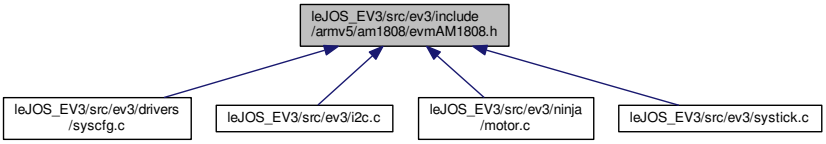
### 4.23.1 Detailed Description

EDMA event enumeration.

## 4.24 leJOS_EV3/src/ev3/include/armv5/am1808/evmAM1808.h File Reference

This file contains the board specific function prototypes for use by applications.

This graph shows which files directly or indirectly include this file:




### Functions

- unsigned int **LCDVersionGet** (void)
- void **UARTPinMuxSetup** (unsigned int instanceNum, unsigned int modemCtrlChoice)
- void **RTCPinMuxSetup** (unsigned int alarmPinReqd)
- void **SPI0CSPinMuxSetup** (unsigned int csPinNum)
- void **SPI1CSPinMuxSetup** (unsigned int csPinNum)
- void **I2CPinMuxSetup** (unsigned int instanceNum)
- void **SPIPinMuxSetup** (unsigned int instanceNum)
- void **ConfigRasterDisplayEnable** (void)
- void **GPIOBank4Pin0PinMuxSetup** (void)
- void SysCfgRegistersUnlock (void)

  *This function unlocks the write-protection of the SYSCFG module.*
- void SysCfgRegistersLock (void)

  *This function imposes write-protection on the SYSCFG module.*
- void **EHRPWM0PinMuxSetup** (void)
- void **EHRPWM1PinMuxSetup** (void)
- void **LIDDDisplayEnable** (void)
- void **McASPPinMuxSetup** (void)
- void **EMACPinMuxSetup** (void)
- void **LIDDPinMuxSetup** (void)
- void **LCDPinMuxSetup** (void)
- void **NANDPinMuxSetup** (void)
- void **EMIFAClkConfig** (void)
- void **VPIFPinMuxSetup** (void)

### 4.24.1 Detailed Description

This file contains the board specific function prototypes for use by applications.

### 4.24.2 Function Documentation

#### 4.24.2.1 void SysCfgRegistersLock ( void )

This function imposes write-protection on the SYSCFG module.

**Returns**

None.

**Note**

On programming the Kick Registers with any value other than the unlock sequence, the SYSCFG module gets locked and its registers cannot be accessed.

#### 4.24.2.2 void SysCfgRegistersUnlock ( void )

This function unlocks the write-protection of the SYSCFG module.

**Returns**

None.

**Note**

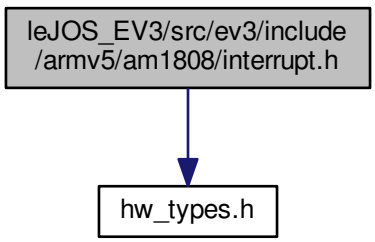
The other registers of the SYSCFG module can be programmed only when an unlock sequence has been written to the Kick Regsiters.

## 4.25 leJOS_EV3/src/ev3/include/armv5/am1808/interrupt.h File Reference

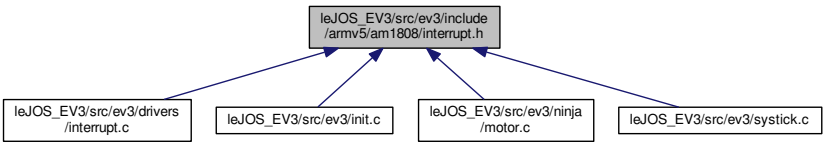
Interrupt related API declarations.

```
#include "hw_types.h"
```

Include dependency graph for interrupt.h:



This graph shows which files directly or indirectly include this file:




## Macros

- #define **SYS_INT_COMMTX** 0
- #define **SYS_INT_COMMRX** 1
- #define **SYS_INT_NINT** 2
- #define **SYS_INT_EVTOUT0** 3
- #define **SYS_INT_EVTOUT1** 4
- #define **SYS_INT_EVTOUT2** 5
- #define **SYS_INT_EVTOUT3** 6
- #define **SYS_INT_EVTOUT4** 7
- #define **SYS_INT_EVTOUT5** 8
- #define **SYS_INT_EVTOUT6** 9
- #define **SYS_INT_EVTOUT7** 10
- #define **SYS_INT_CCINT0** 11
- #define **SYS_INT_CCERRINT** 12
- #define **SYS_INT_TCERRINT0** 13
- #define **SYS_INT_AEMIFINT** 14
- #define **SYS_INT_I2CINT0** 15
- #define **SYS_INT_MMCSDINT0** 16
- #define **SYS_INT_MMCSDINT1** 17
- #define **SYS_INT_PSCINT0** 18
- #define **SYS_INT_RTC** 19
- #define **SYS_INT_SPINT0** 20
- #define **SYS_INT_TINT12_0** 21
- #define **SYS_INT_TINT34_0** 22
- #define **SYS_INT_TINT12_1** 23
- #define **SYS_INT_TINT34_1** 24
- #define **SYS_INT_UARTINT0** 25
- #define **SYS_INT_PROTERR** 26
- #define **SYS_INT_SYSCFG_CHIPINT0** 28
- #define **SYS_INT_SYSCFG_CHIPINT1** 29
- #define **SYS_INT_SYSCFG_CHIPINT2** 30
- #define **SYS_INT_SYSCFG_CHIPINT3** 31
- #define **SYS_INT_TCERRINT1** 32
- #define **SYS_INT_C0_RXTHRESH** 33
- #define **SYS_INT_C0_RX** 34
- #define **SYS_INT_C0_TX** 35
- #define **SYS_INT_C0_MISC** 36
- #define **SYS_INT_C1_RXTHRESH** 37
- #define **SYS_INT_C1_RX** 38
- #define **SYS_INT_C1_TX** 39
- #define **SYS_INT_C1_MISC** 40
- #define **SYS_INT_DDR2MEMERR** 41

- #define **SYS_INT_GPIOB0** 42
- #define **SYS_INT_GPIOB1** 43
- #define **SYS_INT_GPIOB2** 44
- #define **SYS_INT_GPIOB3** 45
- #define **SYS_INT_GPIOB4** 46
- #define **SYS_INT_GPIOB5** 47
- #define **SYS_INT_GPIOB6** 48
- #define **SYS_INT_GPIOB7** 49
- #define **SYS_INT_GPIOB8** 50
- #define **SYS_INT_I2CINT1** 51
- #define **SYS_INT_LCDINT** 52
- #define **SYS_INT_UARTINT1** 53
- #define **SYS_INT_MCASPINT** 54
- #define **SYS_INT_PSCINT1** 55
- #define **SYS_INT_SPINT1** 56
- #define **SYS_INT_UHPI_INT1** 57
- #define **SYS_INT_USB0** 58
- #define **SYS_INT_USB1HC** 59
- #define **SYS_INT_USB1RWAKE** 60
- #define **SYS_INT_UARTINT2** 61
- #define **SYS_INT_EHRPWM0** 63
- #define **SYS_INT_EHRPWM0TZ** 64
- #define **SYS_INT_EHRPWM1** 65
- #define **SYS_INT_EHRPWM1TZ** 66
- #define **SYS_INT_SATA** 67
- #define **SYS_INT_TIMR2_ALL** 68
- #define **SYS_INT_ECAP0** 69
- #define **SYS_INT_ECAP1** 70
- #define **SYS_INT_ECAP2** 71
- #define **SYS_INT_MMCSD** 72
- #define **SYS_INT_SDIO** 73
- #define **SYS_INT_TMR1_CMPINT0** 74
- #define **SYS_INT_TMR1_CMPINT1** 75
- #define **SYS_INT_TMR1_CMPINT2** 76
- #define **SYS_INT_TMR1_CMPINT3** 77
- #define **SYS_INT_TMR1_CMPINT4** 78
- #define **SYS_INT_TMR1_CMPINT5** 79
- #define **SYS_INT_TMR1_CMPINT6** 80
- #define **SYS_INT_TMR1_CMPINT7** 81
- #define **SYS_INT_TMR2_CMPINT0** 82
- #define **SYS_INT_TMR2_CMPINT1** 83
- #define **SYS_INT_TMR2_CMPINT2** 84
- #define **SYS_INT_TMR2_CMPINT3** 85
- #define **SYS_INT_TMR2_CMPINT4** 86
- #define **SYS_INT_TMR2_CMPINT5** 87
- #define **SYS_INT_TMR2_CMPINT6** 88
- #define **SYS_INT_TMR2_CMPINT7** 89
- #define **SYS_INT_ARMCLKSTOPREQ** 90
- #define **SYS_INT_VPIF** 92
- #define **NUM_INTERRUPTS** 101

## Functions

- void IntIRQEnable (void)

  *Enables IRQ in HIER register of AINTC.*
- void IntIRQDisable (void)

  *Disables IRQ in HIER register of AINTC.*
- void IntFIQEnable (void)

  *Enables FIQ in AINTC.*
- void IntFIQDisable (void)

  *Disables FIQ host interrupts in AINTC.*
- void IntAINTCInit (void)

  *This API is used to setup the AINTC. This API shall be called before using the AINTC. All the host interruptsi will be disabled after calling this API. The user shall enable all the interrupts required for processing.*
- void IntGlobalEnable (void)

  *Enables interrupts in GER register of AINTC. FIQ and IRQ will be signaled for processing.*
- void IntGlobalDisable (void)

  *Disables interrupts in GER register of AINTC. No interrupts will be signaled by the AINTC to the ARM core.*
- void IntMasterIRQEnable (void)

  *Enables the processor IRQ only in CPSR. Makes the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- void IntMasterIRQDisable (void)

  *Disables the processor IRQ only in CPSR.Prevents the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- void IntMasterFIQEnable (void)

  *Enables the processor FIQ only in CPSR. Makes the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- void IntMasterFIQDisable (void)

  *Disables the processor FIQ only in CPSR.Prevents the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.*
- void IntSystemEnable (unsigned int intrNum)

  *Enables the sytem interrupt in AINTC. The interrupt will be processed only if it is enabled in AINTC.*
- void IntSystemDisable (unsigned int intrNum)

  *Disables the sytem interrupt in the AINTC.*
- void IntSystemStatusClear (unsigned int intrNum)

  *Clears a sytem interrupt status in AINTC.*
- unsigned char IntChannelGet (unsigned int intrNum)

  *Get the channel number for a system interrupt.*
- unsigned int IntSystemStatusRawGet (unsigned int intrNum)

  *Get the raw status of a system interrupt. This will return 1 if the status is set and return 0 if the status is clear.*
- unsigned int IntSystemStatusEnabledGet (unsigned int intrNum)

  *Get the enabled status of a system interrupt. This will return 1 if the status is set, and return 0 if the status is clear.*
- unsigned int IntMasterStatusGet (void)

  *Returns the status of the interrupts FIQ and IRQ.*
- void IntUnRegister (unsigned int intrNum)

  *Unregisters an interrupt.*
- void IntChannelSet (unsigned int intrNum, unsigned char channel)

  *Set the channel number for a system interrupt. Channel numbers 0-1 are mapped to FIQ and Channel numbers 2-31 are mapped to IRQ of ARM. One or more system interrupt can be mapped to one channel. However, one system interrupt can not be mapped to multiple channels.*
- void IntRegister (unsigned int intrNum, void(*pfnHandler)(void))

  *Registers an interrupt Handler in the interrupt vector table for system interrupts.*
- unsigned char IntDisable (void)

  *Read and save the stasus and Disables the processor IRQ . Prevents the processor to respond to IRQs.*
- void IntEnable (unsigned char status)

  *Restore the processor IRQ only status. This does not affect the set of interrupts enabled/disabled in the AINTC.*

### 4.25.1 Detailed Description

Interrupt related API declarations.

This file contains the API prototypes for configuring AINTC

### 4.25.2 Function Documentation

#### 4.25.2.1 void IntAINTCInit ( void )

This API is used to setup the AINTC. This API shall be called before using the AINTC. All the host interruptsi will be disabled after calling this API. The user shall enable all the interrupts required for processing.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.25.2.2 unsigned char IntChannelGet ( unsigned int *intrNum* )

Get the channel number for a system interrupt.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |

**Returns**

Channel Number.

#### 4.25.2.3 void IntChannelSet ( unsigned int *intrNum,* unsigned char *channel* )

Set the channel number for a system interrupt. Channel numbers 0-1 are mapped to FIQ and Channel numbers 2-31 are mapped to IRQ of ARM. One or more system interrupt can be mapped to one channel. However, one system interrupt can not be mapped to multiple channels.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |
| *channel* | - Channel Number to be set |

**Returns**

None.

#### 4.25.2.4 unsigned char IntDisable ( void )

Read and save the stasus and Disables the processor IRQ . Prevents the processor to respond to IRQs.

**Parameters**

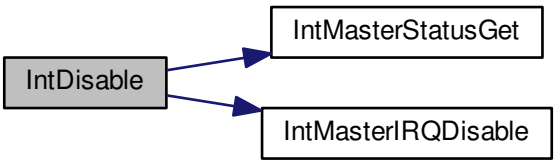
| | |
|---|---|
| *None* | |

**Returns**

Current status of IRQ

Note: This function call shall be done only in previleged mode of ARM

Here is the call graph for this function:




#### 4.25.2.5 void IntEnable ( unsigned char *status* )

Restore the processor IRQ only status. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *The* | status returned by the IntDisable fundtion. |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

Here is the call graph for this function:


#### 4.25.2.6 void IntFIQDisable ( void )

Disables FIQ host interrupts in AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

#### 4.25.2.7 void IntFIQEnable ( void )

Enables FIQ in AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.25.2.8 void IntGlobalDisable ( void )

Disables interrupts in GER register of AINTC. No interrupts will be signaled by the AINTC to the ARM core.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

#### 4.25.2.9 void IntGlobalEnable ( void )

Enables interrupts in GER register of AINTC. FIQ and IRQ will be signaled for processing.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

#### 4.25.2.10 void IntIRQDisable ( void )

Disables IRQ in HIER register of AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.25.2.11 void IntIRQEnable ( void )

Enables IRQ in HIER register of AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None.

#### 4.25.2.12 void IntMasterFIQDisable ( void )

Disables the processor FIQ only in CPSR.Prevents the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.25.2.13 void IntMasterFIQEnable ( void )

Enables the processor FIQ only in CPSR. Makes the processor to respond to FIQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.25.2.14 void IntMasterIRQDisable ( void )

Disables the processor IRQ only in CPSR.Prevents the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.25.2.15 void IntMasterIRQEnable ( void )

Enables the processor IRQ only in CPSR. Makes the processor to respond to IRQs. This does not affect the set of interrupts enabled/disabled in the AINTC.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

None

Note: This function call shall be done only in previleged mode of ARM

#### 4.25.2.16 unsigned int IntMasterStatusGet ( void )

Returns the status of the interrupts FIQ and IRQ.

**Parameters**

| | |
|---|---|
| *None* | |

**Returns**

Status of interrupt as in CPSR.

Note: This function call shall be done only in previleged mode of ARM

#### 4.25.2.17 void IntRegister ( unsigned int *intrNum,* void(*)(void) *fnHandler* )

Registers an interrupt Handler in the interrupt vector table for system interrupts.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |
| *fnHandler* | - Function pointer to the ISR |

Note: When the interrupt occurs for the sytem interrupt number indicated, the control goes to the ISR given as the parameter.

**Returns**

None.

#### 4.25.2.18 void IntSystemDisable ( unsigned int *intrNum* )

Disables the sytem interrupt in the AINTC.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

None

#### 4.25.2.19 void IntSystemEnable ( unsigned int *intrNum* )

Enables the sytem interrupt in AINTC. The interrupt will be processed only if it is enabled in AINTC.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

None.

#### 4.25.2.20 void IntSystemStatusClear ( unsigned int *intrNum* )

Clears a sytem interrupt status in AINTC.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

None

#### 4.25.2.21 unsigned int IntSystemStatusEnabledGet ( unsigned int *intrNum* )

Get the enabled status of a system interrupt. This will return 1 if the status is set, and return 0 if the status is clear.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |

**Returns**

Enabled Interrupt Status.

#### 4.25.2.22 unsigned int IntSystemStatusRawGet ( unsigned int *intrNum* )

Get the raw status of a system interrupt. This will return 1 if the status is set and return 0 if the status is clear.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt number |

**Returns**

Raw Interrupt Status

#### 4.25.2.23 void IntUnRegister ( unsigned int *intrNum* )

Unregisters an interrupt.

**Parameters**

| | |
|---|---|
| *intrNum* | - Interrupt Number |

Note: Once an interrupt is unregistered it will enter infinite loop once an interrupt occurs

**Returns**

None.

## 4.26 leJOS_EV3/src/ev3/include/armv5/cp15.h File Reference

CP15 related function prototypes.

### Functions

- void **CP15ICacheDisable** (void)
- void **CP15DCacheDisable** (void)
- void **CP15ICacheEnable** (void)
- void **CP15DCacheEnable** (void)
- void **CP15DCacheFlush** (void)
- void **CP15DCacheClean** (void)
- void **CP15DCacheCleanFlush** (void)
- void **CP15ICacheFlush** (void)
- void **CP15TtbSet** (unsigned int ttb)
- void **CP15MMUDisable** (void)
- void **CP15MMUEnable** (void)
- void **CP15ICacheFlushBuff** (unsigned int ptr, unsigned int size)
- void **CP15DCacheCleanBuff** (unsigned int bufPtr, unsigned int size)

### 4.26.1 Detailed Description

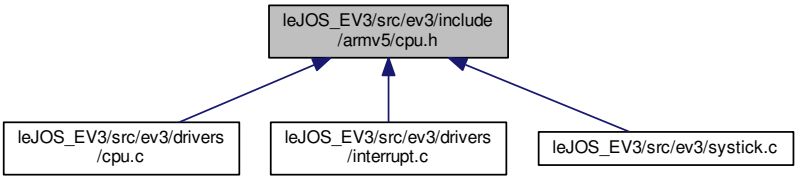
CP15 related function prototypes.

This file contains the API prototypes for configuring CP15

## 4.27 leJOS_EV3/src/ev3/include/armv5/cpu.h File Reference

CPU related function prototypes.

This graph shows which files directly or indirectly include this file:



## Functions

- void CPUSwitchToUserMode (void)

  *This API can be used to switch from any previleged mode of ARM to user mode. After this API is called, the program will continue to operate in non-privileged mode, until any exception occurs. After the exception is serviced, execution will continue in user mode.*
- void CPUSwitchToPrivilegedMode (void)

  *This API can be used to switch from user mode to privileged mode The priviledge mode will be system mode. System mode will share the same resources as user mode, but with privileges.*
- void CPUAbortHandler (void)

  *This API is called when the CPU is aborted or during execution of any undefined instruction. Both IRQ and FIQ will be disabled when the CPU gets an abort and calls this API.*
- void **CPUirqd** (void)
- void **CPUirqe** (void)
- void **CPUfiqd** (void)
- void **CPUfiqe** (void)
- unsigned int **CPUIntStatus** (void)

### 4.27.1 Detailed Description

CPU related function prototypes.

This file contains the API prototypes for configuring CPU

### 4.27.2 Function Documentation

#### 4.27.2.1 void CPUAbortHandler ( void )

This API is called when the CPU is aborted or during execution of any undefined instruction. Both IRQ and FIQ will be disabled when the CPU gets an abort and calls this API.

**Parameters**

| | |
|---|---|
| *None.* | |

**Returns**

None.

Note : The user can perform error handling such as an immediate reset inside this API if required.

#### 4.27.2.2 void CPUSwitchToPrivilegedMode ( void )

This API can be used to switch from user mode to privileged mode The priviledge mode will be system mode. System mode will share the same resources as user mode, but with privileges.

**Parameters**

| | |
|---|---|
| *None.* | |

**Returns**

None.

Note : All the access to system configuration which needs privileged access can be done after calling this API.

#### 4.27.2.3 void CPUSwitchToUserMode ( void )

This API can be used to switch from any previleged mode of ARM to user mode. After this API is called, the program will continue to operate in non-privileged mode, until any exception occurs. After the exception is serviced, execution will continue in user mode.

**Parameters**

| | |
|---|---|
| *None.* | |

**Returns**

None.

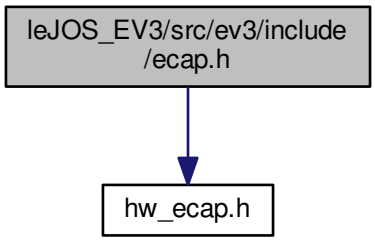
Note : All the access to system configuration which needs privileged access shall be done before this API is called.

## 4.28 leJOS_EV3/src/ev3/include/ecap.h File Reference

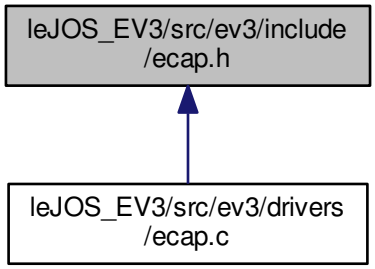
This file contains the function prototypes for the device abstraction layer for ECAP. It also contains some related macro definitions and some files to be included.

```
#include "hw_ecap.h"
```

Include dependency graph for ecap.h:




This graph shows which files directly or indirectly include this file:



## Data Structures

- struct ecapContext

## Macros

- #define **ECAP_CAPTURE_MODE** 1
- #define **ECAP_APWM_MODE** 0
- #define **ECAP_CAPTURE_EVENT_1** 0x08
- #define **ECAP_CAPTURE_EVENT_2** 0x0c
- #define **ECAP_CAPTURE_EVENT_3** 0x10
- #define **ECAP_CAPTURE_EVENT_4** 0x14
- #define **ECAP_CAPTURE_EVENT1_STOP** (0x00 << ECAP_ECCTL2_STOP_WRAP_SHIFT)
- #define **ECAP_CAPTURE_EVENT2_STOP** (0x01 << ECAP_ECCTL2_STOP_WRAP_SHIFT)
- #define **ECAP_CAPTURE_EVENT3_STOP** (0x02 << ECAP_ECCTL2_STOP_WRAP_SHIFT)
- #define **ECAP_CAPTURE_EVENT4_STOP** (0x03 << ECAP_ECCTL2_STOP_WRAP_SHIFT)
- #define **ECAP_APWM_ACTIVE_HIGH** 0
- #define **ECAP_APWM_ACTIVE_LOW** 1
- #define **ECAP_COUNTER_STOP** 0
- #define **ECAP_COUNTER_FREE_RUNNING** 1
- #define **ECAP_SYNC_IN_DISABLE** (0 << ECAP_ECCTL2_SYNCI_EN_SHIFT)
- #define **ECAP_ENABLE_Counter** (1 << ECAP_ECCTL2_SYNCI_EN_SHIFT)
- #define **ECAP_SYNC_IN** (00 << ECAP_ECCTL2_SYNCO_SEL_SHIFT)
- #define **ECAP_PRD_EQ** (01 << ECAP_ECCTL2_SYNCO_SEL_SHIFT)
- #define **ECAP_SYNC_OUT_DISABLE** (10 << ECAP_ECCTL2_SYNCO_SEL_SHIFT)
- #define **ECAP_CEVT1_INT** ECAP_ECEINT_CEVT1
- #define **ECAP_CEVT2_INT** ECAP_ECEINT_CEVT2
- #define **ECAP_CEVT3_INT** ECAP_ECEINT_CEVT3
- #define **ECAP_CEVT4_INT** ECAP_ECEINT_CEVT4
- #define **ECAP_CNTOVF_INT** ECAP_ECEINT_CTROVF
- #define **ECAP_PRDEQ_INT** ECAP_ECEINT_CTR_PRD
- #define **ECAP_CMPEQ_INT** ECAP_ECEINT_CTR_CMP
- #define **ECAP_GLOBAL_INT** ECAP_ECFLG_INT
- #define **ECAP_SMART_STAND_BY_WAKE_UP** 3
- #define **ECAP_FORCE_STAND_BY** 0
- #define **ECAP_SMART_STAND_BY** 2
- #define **ECAP_NO_STAND_BY** 1
- #define **ECAP_SMART_IDLE_WAKE_UP** 3
- #define **ECAP_SMART_IDLE_MODE** 2
- #define **ECAP_FORCE_IDLE_MODE** 0
- #define **ECAP_NO_IDLE_MODE** 1

## Typedefs

- typedef struct ecapContext **ECAPCONTEXT**

## Functions

- void **ECAPClockStop** (unsigned int baseAdd)
- void ECAPClockEnable (unsigned int baseAdd)

  *This functions enables clock for ECAP module in PWMSS subsystem.*
- void ECAPOneShotREARM (unsigned int baseAdd)

  *This function configures ECAP to One-Short Re-arming.*
- void ECAPGlobalIntEnable (unsigned int baseAdd)

  *This function enables the generation of interrupts if any of event interrupt are enable and corresponding event interrupt flag is set.*
- void ECAPContinousModeConfig (unsigned int baseAdd)

  *This function configures ECAP to Continuous mode.*
- void ECAPCaptureLoadingEnable (unsigned int baseAdd)

  *This function enables capture loading.*
- void ECAPCaptureLoadingDisable (unsigned int baseAdd)

  *This function disables capture loading.*
- unsigned int ECAPPeripheralIdGet (unsigned int baseAdd)

  *This function returns the peripheral ID.*
- unsigned int **ECAPClockStopStatusGet** (unsigned int baseAdd)
- unsigned int ECAPClockEnableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is enabled or not.*
- void ECAPIntEnable (unsigned int baseAdd, unsigned int flag)

  *This function enables the specified interrups.*
- void ECAPIntDisable (unsigned int baseAdd, unsigned int flag)

  *This function disables the specified interrups.*
- void ECAPIntStatusClear (unsigned int baseAdd, unsigned int flag)

  *This function clears of the status specified interrups.*
- void ECAPCounterControl (unsigned int baseAdd, unsigned int flag)

  *This function configures counter to stop or free running based on its input argument flag.*
- void ECAPCounterConfig (unsigned int baseAdd, unsigned int countVal)

  *This function configures the counter register which is used as Capture Time base.*
- unsigned int ECAPIntStatus (unsigned int baseAdd, unsigned int flag)

  *This function returns the status specified interrups.*
- void ECAPAPWMPolarityConfig (unsigned int baseAdd, unsigned int flag)

  *This function configures output polarity for APWM output.*
- void ECAPSyncInOutSelect (unsigned int baseAdd, unsigned int syncIn, unsigned int syncOut)

  *This function configures Sync-In and Sync-Out.*
- void ECAPPrescaleConfig (unsigned int baseAdd, unsigned int prescale)

  *This function configures prescale value.*
- void ECAPOneShotModeConfig (unsigned int baseAdd, unsigned int stopVal)

  *This function configures ECAP to One-shot mode and also stop value for this mode.*
- void ECAPAPWMCaptureConfig (unsigned int baseAdd, unsigned int compareVal, unsigned int periodVal)

  *When ECAP module is configured in APWM mode capture 1 and capture 2 registers are used as period and compare register.This function configures compare and period values to this register.*
- void ECAPAPWMShadowCaptureConfig (unsigned int baseAdd, unsigned int compareVal, unsigned int periodVal)

  *This function configures the Shadow register.*
- void ECAPOperatingModeSelect (unsigned int baseAdd, unsigned int modeSelect)

  *This function configures ecapture module to operate in capture mode or in APWM mode.*
- unsigned int ECAPTimeStampRead (unsigned int baseAdd, unsigned int capEvtFlag)

  *This function returns time-stamp for a given capture event.*
- void ECAPCounterPhaseValConfig (unsigned int baseAdd, unsigned int cntPhaseVal)

*This function configures the counter phase value.*

- void ECAPCapeEvtPolarityConfig (unsigned int baseAdd, unsigned int capEvt1pol, unsigned int capEvt2pol, unsigned int capEvt3pol, unsigned int capEvt4pol)

  *This function configures Capture Event polarity.*
- void ECAPCaptureEvtCntrRstConfig (unsigned int baseAdd, unsigned int CounterRst1, unsigned int CounterRst2, unsigned int CounterRst3, unsigned int CounterRst4)

  *This function enables reset of the counters upon Capture Events.*
- void ECAPClockDisable (unsigned int baseAdd)

  *This functions disables clock for ECAP module in PWMSS subsystem.*
- unsigned int ECAPClockDisableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is disabled or not.*
- void EcapContextSave (unsigned int ecapBase, unsigned int pwmssBase, ECAPCONTEXT *contextPtr)

  *This API can be used to save the register context of ECAP.*
- void EcapContextRestore (unsigned int ecapBase, unsigned int pwmssBase, ECAPCONTEXT *contextPtr)

  *This API can be used to restore the register context of ECAP.*

### 4.28.1 Detailed Description

This file contains the function prototypes for the device abstraction layer for ECAP. It also contains some related macro definitions and some files to be included.

### 4.28.2 Function Documentation

#### 4.28.2.1 void ECAPAPWMCaptureConfig ( unsigned int *baseAdd,* unsigned int *compareVal,* unsigned int *periodVal* )

When ECAP module is configured in APWM mode capture 1 and capture 2 registers are used as period and compare register.This function configures compare and period values to this register.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *compareVal* | It is the Compare value to be configured. |
| *periodVal* | It is the Period value to be configured. |

**Returns**

None.

#### 4.28.2.2 void ECAPAPWMPolarityConfig ( unsigned int *baseAdd,* unsigned int *flag* )

This function configures output polarity for APWM output.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which determines the output polarity for APWM output.<br>flag can take following macros.<br>ECAP_APWM_ACTIVE_HIGH.<br>ECAP_APWM_ACTIVE_LOW. |

**Returns**

None.

#### 4.28.2.3 void ECAPAPWMShadowCaptureConfig ( unsigned int *baseAdd,* unsigned int *compareVal,* unsigned int *periodVal* )

This function configures the Shadow register.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *compareVal* | It is the Compare value to be configured. |
| *periodVal* | It is the Period value to be configured. |

**Returns**

None.

#### 4.28.2.4 void ECAPCapeEvtPolarityConfig ( unsigned int *baseAdd,* unsigned int *capEvt1pol,* unsigned int *capEvt2pol,* unsigned int *capEvt3pol,* unsigned int *capEvt4pol* )

This function configures Capture Event polarity.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *capEvt1pol* | It determines whether Capture Event1 has to be generated on rising edge or falling edge of pulse. |
| *capEvt2pol* | It determines whether Capture Event2 has to be generated on rising edge or falling edge of pulse. |
| *capEvt3pol* | It determines whether Capture Event3 has to be generated on rising edge or falling edge of pulse. |
| *capEvt4pol* | It determines whether Capture Event4 has to be generated on rising edge or falling edge of pulse. |

0 - falling edge 1 - rising edge

**Returns**

None.

#### 4.28.2.5 void ECAPCaptureEvtCntrRstConfig ( unsigned int *baseAdd,* unsigned int *CounterRst1,* unsigned int *CounterRst2,* unsigned int *CounterRst3,* unsigned int *CounterRst4* )

This function enables reset of the counters upon Capture Events.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *CounterRst1* | It determines whether counter has to be reset upon Capture Event1. |
| *CounterRst2* | It determines whether counter has to be reset upon Capture Event2. |

| | |
|---|---|
| *CounterRst3* | It determines whether counter has to be reset upon Capture Event3. |
| *CounterRst4* | It determines whether counter has to be reset upon Capture Event4. |

0 - Don't reset counter upon capture event.
1 - Reset upon counter capture event.

**Returns**

None.

#### 4.28.2.6 void ECAPCaptureLoadingDisable ( unsigned int *baseAdd* )

This function disables capture loading.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.

#### 4.28.2.7 void ECAPCaptureLoadingEnable ( unsigned int *baseAdd* )

This function enables capture loading.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.

#### 4.28.2.8 void ECAPClockDisable ( unsigned int *baseAdd* )

This functions disables clock for ECAP module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.28.2.9 unsigned int ECAPClockDisableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is disabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is disabled. return's '0' if clocked is not disabled.

#### 4.28.2.10 void ECAPClockEnable ( unsigned int *baseAdd* )

This functions enables clock for ECAP module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.28.2.11 unsigned int ECAPClockEnableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is enabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is enabled. return's '0' if clocked is not enabled.

#### 4.28.2.12 void EcapContextRestore ( unsigned int *ecapBase,* unsigned int *pwmssBase,* ECAPCONTEXT * *contextPtr* )

This API can be used to restore the register context of ECAP.

**Parameters**

| | |
|---|---|
| *ecapBase* | Base address of ECAP instance |
| *pwmssBase* | Base address of the PWM subsystem |
| *contextPtr* | Pointer to the structure where ECAP register context need to be saved |

**Returns**

None

#### 4.28.2.13 void EcapContextSave ( unsigned int *ecapBase,* unsigned int *pwmssBase,* ECAPCONTEXT * *contextPtr* )

This API can be used to save the register context of ECAP.

**Parameters**

| | |
|---|---|
| *ecapBase* | Base address of ECAP instance |

| | |
|---|---|
| *pwmssBase* | Base address of the PWM subsystem |
| *contextPtr* | Pointer to the structure where ECAP register context need to be saved |

**Returns**

None

#### 4.28.2.14 void ECAPContinousModeConfig ( unsigned int *baseAdd* )

This function configures ECAP to Continuous mode.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.
This API is valid only if ECAP is configured to Capture Mode.It has no significance when ECAP is configured in APWM mode.

#### 4.28.2.15 void ECAPCounterConfig ( unsigned int *baseAdd,* unsigned int *countVal* )

This function configures the counter register which is used as Capture Time base.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *countVal* | It is counter value to be configured. |

**Returns**

None.

#### 4.28.2.16 void ECAPCounterControl ( unsigned int *baseAdd,* unsigned int *flag* )

This function configures counter to stop or free running based on its input argument flag.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which determine counter to be configured to stop or free running.<br>flag can take following macros.<br>ECAP_COUNTER_STOP.<br>ECAP_COUNTER_FREE_RUNNING. |

**Returns**

None.

#### 4.28.2.17 void ECAPCounterPhaseValConfig ( unsigned int *baseAdd,* unsigned int *cntPhaseVal* )

This function configures the counter phase value.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *cntPhaseVal* | It is the counter phase value to be programmed for phase lag/lead. |

**Returns**

None.

#### 4.28.2.18 void ECAPGlobalIntEnable ( unsigned int *baseAdd* )

This function enables the generation of interrupts if any of event interrupt are enable and corresponding event interrupt flag is set.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.

#### 4.28.2.19 void ECAPIntDisable ( unsigned int *baseAdd,* unsigned int *flag* )

This function disables the specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the interrupts to be disabled.<br>flag can take following macros. |

ECAP_CEVT1_INT - Enable Capture Event 1 interrupt.
ECAP_CEVT2_INT - Enable Capture Event 2 interrupt.
ECAP_CEVT3_INT - Enable Capture Event 3 interrupt.
ECAP_CEVT4_INT - Enable Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Enable Counter Overflow interrupt.
ECAP_PRDEQ_INT - Enable Period equal interrupt.
ECAP_CMPEQ_INT - Enable Compare equal interrupt.

**Returns**

None.

#### 4.28.2.20 void ECAPIntEnable ( unsigned int *baseAdd,* unsigned int *flag* )

This function enables the specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the interrupts to be enabled.<br>flag can take following macros |

ECAP_CEVT1_INT - Enable Capture Event 1 interrupt.

ECAP_CEVT2_INT - Enable Capture Event 2 interrupt.
ECAP_CEVT3_INT - Enable Capture Event 3 interrupt.
ECAP_CEVT4_INT - Enable Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Enable Counter Overflow interrupt.
ECAP_PRDEQ_INT - Enable Period equal interrupt.
ECAP_CMPEQ_INT - Enable Compare equal interrupt.

**Returns**

None.

#### 4.28.2.21 unsigned int ECAPIntStatus ( unsigned int *baseAdd,* unsigned int *flag* )

This function returns the status specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the status of interrupts to be returned.<br>flag can take following macros. |

ECAP_CEVT1_INT - Status of Capture Event 1 interrupt.
ECAP_CEVT2_INT - Status of Capture Event 2 interrupt.
ECAP_CEVT3_INT - Status of Capture Event 3 interrupt.
ECAP_CEVT4_INT - Status of Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Status of Counter Overflow interrupt.
ECAP_PRDEQ_INT - Status of Period equal interrupt.
ECAP_CMPEQ_INT - Status of Compare equal interrupt.
ECAP_GLOBAL_INT - Global interrupt status.

**Returns**

Status of the specified interrupts.

#### 4.28.2.22 void ECAPIntStatusClear ( unsigned int *baseAdd,* unsigned int *flag* )

This function clears of the status specified interrups.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *flag* | It is the value which specifies the status of interrupts to be cleared.<br>flag can take following macros. |

ECAP_CEVT1_INT - Status of Capture Event 1 interrupt.
ECAP_CEVT2_INT - Status of Capture Event 2 interrupt.
ECAP_CEVT3_INT - Status of Capture Event 3 interrupt.
ECAP_CEVT4_INT - Status of Capture Event 4 interrupt.
ECAP_CNTOVF_INT - Status of Counter Overflow interrupt.
ECAP_PRDEQ_INT - Status of Period equal interrupt.
ECAP_CMPEQ_INT - Status of Compare equal interrupt.

**Returns**

None.

#### 4.28.2.23 void ECAPOneShotModeConfig ( unsigned int *baseAdd,* unsigned int *stopVal* )

This function configures ECAP to One-shot mode and also stop value for this mode.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *stopVal* | It is the number of captures allowed to occur before Capture register(1-4) are frozen.<br>stopVal can take following macros.<br>ECAP_CAPTURE_EVENT1_STOP - stop after Capture Event 1<br>. ECAP_CAPTURE_EVENT2_STOP - stop after Capture Event 2<br>. ECAP_CAPTURE_EVENT3_STOP - stop after Capture Event 3<br>. ECAP_CAPTURE_EVENT4_STOP - stop after Capture Event 4<br>. |

**Returns**

None.
This API is valid only if ECAP is configured to Capture Mode.It has no significance when ECAP is configured in APWM mode.

#### 4.28.2.24 void ECAPOneShotREARM ( unsigned int *baseAdd* )

This function configures ECAP to One-Short Re-arming.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

None.
When this API is invoked following things happen.

1. Resets Mod4 counter to zero.
2. Un-freezes the Mod4 counter.
3. Enables capture register loads.

#### 4.28.2.25 void ECAPOperatingModeSelect ( unsigned int *baseAdd,* unsigned int *modeSelect* )

This function configures ecapture module to operate in capture mode or in APWM mode.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *modeSelect* | It is the value which determines whether ecapture module to operate in capture mode or in APWM mode.<br>modeSelect can take following macros. |

ECAP_CAPTURE_MODE - Capture Mode.
ECAP_APWM_MODE - APWM Mode.

**Returns**

None.

#### 4.28.2.26 unsigned int ECAPPeripheralIdGet ( unsigned int *baseAdd* )

This function returns the peripheral ID.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |

**Returns**

Peripheral ID.

#### 4.28.2.27 void ECAPPrescaleConfig ( unsigned int *baseAdd,* unsigned int *prescale* )

This function configures prescale value.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *prescale* | It is the value which is used to prescale the incoming input. |

prescale can take any integer value between 0 and 62

**Returns**

None.

#### 4.28.2.28 void ECAPSyncInOutSelect ( unsigned int *baseAdd,* unsigned int *syncIn,* unsigned int *syncOut* )

This function configures Sync-In and Sync-Out.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *syncIn* | It is the value which determines whether to disable syncIn or to enable counter to be loaded from CNTPHS register upon a SYNCI signal.<br>syncIn can take following macros.<br>ECAP_SYNC_IN_DISABLE.<br>ECAP_ENABLE_COUNTER - Enables counter to load from CNTPHS register upon SYNCI signal. |
| *syncOut* | It is the value which select type of syncOut signal (i.e select syncIn event to be the Sync-Out signal, select PRD_eq event to be Sync-Out signal).<br><br>`syncOut can take following macros.\n`<br><br>`ECAP_SYNC_IN - Select syncIn event to be the Sync-Out signal.\n`<br>`ECAP_PRD_EQ  - Select PRD_eq event to be Sync-Out signal.\n`<br>`ECAP_SYNC_OUT_DISABLE - Disable syncOut signal.\n` |

**Returns**

None.

#### 4.28.2.29 unsigned int ECAPTimeStampRead ( unsigned int *baseAdd*, unsigned int *capEvtFlag* )

This function returns time-stamp for a given capture event.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the ECAP instance used. |
| *capEvtFlag* | It is the value which determines for which capture event time-stam has to returned. |

capEvtFlag can take following macros.

ECAP_CAPTURE_EVENT_1 - Capture Event 1.
ECAP_CAPTURE_EVENT_2 - Capture Event 2.
ECAP_CAPTURE_EVENT_3 - Capture Event 3.
ECAP_CAPTURE_EVENT_4 - Capture Event 4.

**Returns**

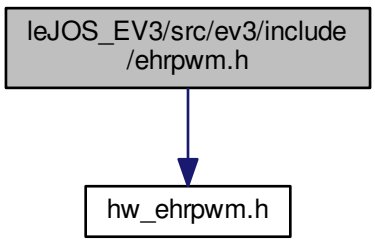
Returns the time-stamp for given capure event.

## 4.29 leJOS_EV3/src/ev3/include/ehrpwm.h File Reference

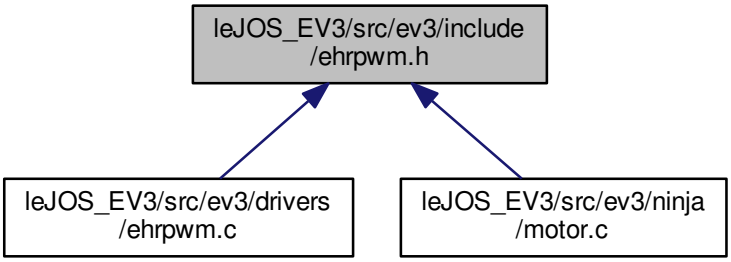
This file contains the Macros and API prototypes for ehrpwm driver.

```
#include "hw_ehrpwm.h"
```

Include dependency graph for ehrpwm.h:



This graph shows which files directly or indirectly include this file:




## Macros

- #define **EHRPWM_PRD_LOAD_SHADOW_MASK** EHRPWM_TBCTL_PRDLD
- #define **EHRPWM_COUNTER_MODE_MASK** EHRPWM_TBCTL_CTRMODE
- #define **EHRPWM_COUNT_UP**
- #define **EHRPWM_COUNT_DOWN**
- #define **EHRPWM_COUNT_UP_DOWN**
- #define **EHRPWM_COUNT_STOP**
- #define **EHRPWM_SYNC_ENABLE** EHRPWM_TBCTL_PHSEN
- #define **EHRPWM_SYNCOUT_MASK** EHRPWM_TBCTL_SYNCOSEL
- #define **EHRPWM_SYNCOUT_SYNCIN**
- #define **EHRPWM_SYNCOUT_COUNTER_EQUAL_ZERO**
- #define **EHRPWM_SYNCOUT_COUNTER_EQUAL_COMPAREB**
- #define **EHRPWM_SYNCOUT_DISABLE**
- #define **EHRPWM_SHADOW_WRITE_ENABLE** 0x0
- #define **EHRPWM_SHADOW_WRITE_DISABLE** 0x1
- #define **EHRPWM_STOP_AFTER_NEXT_TB_INCREMENT** (0x0 << EHRPWM_TBCTL_FREE_SOFT_↩ SHIFT)
- #define **EHRPWM_STOP_AFTER_A_COMPLETE_CYCLE** (0x1 << EHRPWM_TBCTL_FREE_SOFT_↩ SHIFT)
- #define **EHRPWM_FREE_RUN** (0x2 << EHRPWM_TBCTL_FREE_SOFT_SHIFT)
- #define **EHRPWM_TBCTL_CLKDIV_1** (0x0001u)
- #define **EHRPWM_TBCTL_CLKDIV_2** (0x0002u)
- #define **EHRPWM_TBCTL_CLKDIV_4** (0x0004u)
- #define **EHRPWM_TBCTL_CLKDIV_8** (0x0008u)
- #define **EHRPWM_TBCTL_CLKDIV_16** (0x0010u)
- #define **EHRPWM_TBCTL_CLKDIV_32** (0x0020u)
- #define **EHRPWM_TBCTL_CLKDIV_64** (0x0040u)
- #define **EHRPWM_TBCTL_CLKDIV_128** (0x0080u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_1** (0x0001u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_2** (0x0002u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_4** (0x0004u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_6** (0x0006u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_8** (0x0008u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_10** (0x000Au)
- #define **EHRPWM_TBCTL_HSPCLKDIV_12** (0x000Cu)
- #define **EHRPWM_TBCTL_HSPCLKDIV_14** (0x000Eu)

- #define **EHRPWM_COUNT_DOWN_AFTER_SYNC** 0x0
- #define **EHRPWM_COUNT_UP_AFTER_SYNC** 0x1
- #define **EHRPWM_SHADOW_A_EMPTY** (0x0 << EHRPWM_CMPCTL_SHDWAFULL_SHIFT)
- #define **EHRPWM_SHADOW_A_FULL** (EHRPWM_CMPCTL_SHDWAFULL)
- #define **EHRPWM_SHADOW_B_EMPTY** (0x0 << EHRPWM_CMPCTL_SHDWBFULL_SHIFT)
- #define **EHRPWM_SHADOW_B_FULL** (EHRPWM_CMPCTL_SHDWBFULL)
- #define **EHRPWM_CMPCTL_NOT_OVERWR_SH_FL** 0x0
- #define **EHRPWM_CMPCTL_OVERWR_SH_FL** 0x1
- #define **EHRPWM_COMPB_LOAD_MASK** EHRPWM_CMPCTL_LOADBMODE
- #define **EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO**
- #define **EHRPWM_COMPB_LOAD_COUNT_EQUAL_PERIOD**
- #define **EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD**
- #define **EHRPWM_COMPB_NO_LOAD**
- #define **EHRPWM_COMPA_LOAD_MASK** EHRPWM_CMPCTL_LOADAMODE
- #define **EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO**
- #define **EHRPWM_COMPA_LOAD_COUNT_EQUAL_PERIOD**
- #define **EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD**
- #define **EHRPWM_COMPA_NO_LOAD**
- #define **EHRPWM_CHP_DUTY_12_5_PER** EHRPWM_PCCTL_CHPDUTY_1DIV8
- #define **EHRPWM_CHP_DUTY_25_PER** EHRPWM_PCCTL_CHPDUTY_2DIV8
- #define **EHRPWM_CHP_DUTY_37_5_PER** EHRPWM_PCCTL_CHPDUTY_3DIV8
- #define **EHRPWM_CHP_DUTY_50_PER** EHRPWM_PCCTL_CHPDUTY_4DIV8
- #define **EHRPWM_CHP_DUTY_62_5_PER** EHRPWM_PCCTL_CHPDUTY_5DIV8
- #define **EHRPWM_CHP_DUTY_75_PER** EHRPWM_PCCTL_CHPDUTY_6DIV8
- #define **EHRPWM_CHP_DUTY_87_5_PER** EHRPWM_PCCTL_CHPDUTY_7DIV8
- #define **EHRPWM_TZ_ONESHOT** 0x0
- #define **EHRPWM_TZ_CYCLEBYCYCLE** 0x1
- #define **EHRPWM_TZ_ONESHOT_CLEAR** (EHRPWM_TZCLR_OST | EHRPWM_TZCLR_INT)
- #define **EHRPWM_TZ_CYCLEBYCYCLE_CLEAR** (EHRPWM_TZCLR_CBC | EHRPWM_TZCLR_INT)
- #define **ECAP** 0x01
- #define **EPWM** 0x02
- #define **EQEP** 0x03

## Typedefs

- typedef char **bool**

## Functions

- void EHRPWMTimebaseClkConfig (unsigned int baseAddr, unsigned int tbClk, unsigned int moduleClk)

  *This API configures the clock divider of the Time base module. The clock divider can be calculated using the equation TBCLK = SYSCLKOUT/(HSPCLKDIV LKDIV)*
- void EHRPWMPWMOpFreqSet (unsigned int baseAddr, unsigned int tbClk, unsigned int pwmFreq, unsigned int counterDir, bool enableShadowWrite)

  *This API configures the PWM Frequency/Period. The period count determines the period of the final output waveform. For the given period count, in the case of UP and DOWN counter the count value will be loaded as is. In the case of UP_DOWN counter the count is halfed.*
- void EHRPWMTBEmulationModeSet (unsigned int baseAddr, unsigned int mode)

  *This API configures emulation mode. This setting determines the behaviour of Timebase during emulation (debugging).*
- void EHRPWMTimebaseSyncEnable (unsigned int baseAddr, unsigned int tbPhsValue, unsigned int phs↩ CountDir)

  *This API enables the synchronization. When a sync-in event is generated the couter is reloaded with the new value. After sync the counter will use the new value.*

- void EHRPWMTimebaseSyncDisable (unsigned int baseAddr)

  *This API disables the synchronization. Even if sync-in event occurs the count value will not be reloaded.*
- void EHRPWMTriggerSWSync (unsigned int baseAddr)

  *This API generates sw forced sync pulse. This API can be used for testing. When this API is called sync-in will be generated.*
- void EHRPWMSyncOutModeSet (unsigned int baseAddr, unsigned int syncOutMode)

  *This API selects the output sync source. It determines on which of the following event sync-out has to be generated.*
- void EHRPWMWriteTBCount (unsigned int baseAddr, unsigned int tbCount)

  *This API loads the TB counter. The new value is taken immediately.*
- unsigned int EHRPWMReadTBCount (unsigned int baseAddr)

  *This API gets the TB counter current value. The count operation is not affected by the read.*
- unsigned int EHRPWMTBStatusGet (unsigned int baseAddr, unsigned int tbStatusMask)

  *This API gets the TB status as indicated by the tbStatusMask parameter.*
- void EHRPWMTBClearStatus (unsigned int baseAddr, unsigned int tbStatusMask)

  *This API clears the TB status bits indicated by the tbStatusMask parameter.*
- bool EHRPWMLoadCMPA (unsigned int baseAddr, unsigned int CMPAVal, bool enableShadowWrite, unsigned int ShadowToActiveLoadTrigger, bool OverwriteShadowFull)

  *This API loads the CMPA value. When CMPA value equals the counter value, then an event is generated both in the up direction and down direction.*
- bool EHRPWMLoadCMPB (unsigned int baseAddr, unsigned int CMPBVal, bool enableShadowWrite, unsigned int ShadowToActiveLoadTrigger, bool OverwriteShadowFull)

  *This API loads the CMPB value. When CMPB value equals the counter value, then an event is generated both in the up direction and down direction.*
- void EHRPWMConfigureAQActionOnA (unsigned int baseAddr, unsigned int zero, unsigned int period, unsigned int CAUp, unsigned int CADown, unsigned int CBUp, unsigned int CBDown, unsigned int SWForced)

  *This API configures the action to be taken on A by the Action qualifier module upon receiving the events. This will determine the output waveform.*
- void EHRPWMConfigureAQActionOnB (unsigned int baseAddr, unsigned int zero, unsigned int period, unsigned int CAUp, unsigned int CADown, unsigned int CBUp, unsigned int CBDown, unsigned int SWForced)

  *his API configures the action to be taken on B by the Action qualifier module upon receiving the events. This will determine the output waveform.*
- void EHRPWMSWForceA (unsigned int baseAddr)

  *This API triggers the SW forced event on A. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.*
- void EHRPWMSWForceB (unsigned int baseAddr)

  *This API triggers the SW forced event on B. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.*
- void EHRPWMAQContSWForceOnA (unsigned int baseAddr, unsigned int forceVal, unsigned int activeReg↩ ReloadMode)

  *This API forces a value continuously on A. The output can be forced to low or high.*
- void EHRPWMAQContSWForceOnB (unsigned int baseAddr, unsigned int forceVal, unsigned int activeReg↩ ReloadMode)

  *This API forces a value continuously on B. The output can be forced to low or high.*
- void EHRPWMDBSourceSelect (unsigned int baseAddr, unsigned int DBgenSource)

  *This API selects the source for delay blocks in dead band sub-module. The Dead band generator has two sub-modules, one for raising edge delay and the other for falling edge delay. This can be configured when a delay is need between two signals during signal change. The dead band generator is usefull in full-inverters.*
- void EHRPWMDBPolaritySelect (unsigned int baseAddr, unsigned int DBgenPol)

  *This API selects the polarity. This allows to selectively invert one of the delayed signals before it is sent out of the dead-band sub-module.*
- void EHRPWMDBOutput (unsigned int baseAddr, unsigned int DBgenOpMode)

  *This API selects output mode. This allows to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.*
- void EHRPWMDBConfigureRED (unsigned int baseAddr, unsigned int raisingEdgeDelay)

*This API sets the raising edge delay.*

- void EHRPWMDBConfigureFED (unsigned int baseAddr, unsigned int fallingEdgeDelay)

  *This API sets the Falling edge delay.*
- void EHRPWMConfigureChopperDuty (unsigned int baseAddr, unsigned int dutyCycle)

  *This API configures the chopper duty cyce. In Chopper sub-module the PWM signal is modulated with a carrier signal. Th duty cycle of the carrier signal is configured with this API.*
- void EHRPWMConfigureChopperFreq (unsigned int baseAddr, unsigned int freqDiv)

  *This API configures the chopper frequency. In chopper sub-module the PWM signal is modulated with a carrier signal. The frequency of the carrier signal is configured with this API.*
- void EHRPWMConfigureChopperOSPW (unsigned int baseAddr, unsigned int OSPWCycles)

  *This API configures one shot pulse width. The chopper module is useful in switching operations for pulse transformers. The one-shot block provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on.*
- void EHRPWMChopperEnable (unsigned int baseAddr)

  *This API enables the PWM chopper sub-module.*
- void EHRPWMChopperDisable (unsigned int baseAddr)

  *This API disables the PWM chopper sub-module. This will cause the chopper module to be by-passed.*
- void EHRPWMTZTripEventEnable (unsigned int baseAddr, bool osht_CBC)

  *This API enables the trip event. The trip signals indicates external fault, and the ePWM outputs can be programmed to respond accordingly when faults occur.*
- void EHRPWMTZTripEventDisable (unsigned int baseAddr, bool osht_CBC)

  *This API disable the trip event. The trip events will be ignored.*
- void EHRPWMTZForceAOnTrip (unsigned int baseAddr, unsigned int opValue)

  *This API configures the o/p on A when a trip event is recognized. The output can be set to high or low or high impedence.*
- void EHRPWMTZForceBOnTrip (unsigned int baseAddr, unsigned int opValue)

  *This API configures the o/p on B when a trip event is recognised. The output can be set to high or low or high impedence.*
- void EHRPWMTZIntEnable (unsigned int baseAddr, bool osht_CBC)

  *This API enables the trip interrupt. When trip event occurs the sub-module can be configured to interrupt CPU.*
- void EHRPWMTZIntDisable (unsigned int baseAddr, bool osht_CBC)

  *This API disables the trip interrupt.*
- unsigned int EHRPWMTZFlagGet (unsigned int baseAddr, unsigned int flagToRead)

  *This API returns the flag status requested.*
- void EHRPWMTZFlagClear (unsigned int baseAddr, unsigned int flagToClear)

  *This API clears the flag.*
- void EHRPWMTZSWFrcEvent (unsigned int baseAddr, bool osht_CBC)

  *This API enables to generate SW forced trip.*
- void EHRPWMETIntDisable (unsigned int baseAddr)

  *This API disables the interrupt.*
- void EHRPWMETIntEnable (unsigned int baseAddr)

  *This API enables the interrupt.*
- void EHRPWMETIntSourceSelect (unsigned int baseAddr, unsigned int selectInt)

  *This API selects the interrupt source.*
- void EHRPWMETIntPrescale (unsigned int baseAddr, unsigned int prescale)

  *This API prescales the event on which interrupt is to be generated.*
- bool EHRPWMETEventCount (unsigned int baseAddr)

  *This API returns the number of events occured.*
- bool EHRPWMETIntStatus (unsigned int baseAddr)

  *This API returns the interrupt status.*
- void EHRPWMETIntClear (unsigned int baseAddr)

  *This API clears the interrupt.*

- void EHRPWMETIntSWForce (unsigned int baseAddr)

  *This API forces interrupt to be generated.*
- void EHRPWMLoadTBPHSHR (unsigned int baseAddr, unsigned int TBPHSHRVal)

  *This API loads the HR PHS value.*
- void EHRPWMLoadCMPAHR (unsigned int baseAddr, unsigned int CMPAHRVal, unsigned int ShadowTo↩ ActiveLoadTrigger)

  *This API loads CMPAHR value.*
- void EHRPWMConfigHR (unsigned int baseAddr, unsigned int ctrlMode, unsigned int MEPCtrlEdge)

  *This API configures control mode and edge mode. In also enables the HR sub-module.*
- void EHRPWMHRDisable (unsigned int baseAddr)

  *This API disables the HR sub-module.*
- void EHRPWMClockEnable (unsigned int baseAdd)

  *This functions enables clock for EHRPWM module in PWMSS subsystem.*
- void EHRPWMClockDisable (unsigned int baseAdd)

  *This functions enables clock for EHRPWM module in PWMSS subsystem.*
- unsigned int EHRPWMClockEnableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is enabled or not.*
- unsigned int EHRPWMClockDisableStatusGet (unsigned int baseAdd)

  *This functions determines whether clock is disabled or not.*

### 4.29.1 Detailed Description

This file contains the Macros and API prototypes for ehrpwm driver.

==========

### 4.29.2 Macro Definition Documentation

#### 4.29.2.1 #define EHRPWM_COMPA_LOAD_COUNT_EQUAL_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_TBCTRPRD << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

#### 4.29.2.2 #define EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_TBCTRZERO << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

#### 4.29.2.3 #define EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_ZEROORPRD << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

#### 4.29.2.4 #define EHRPWM_COMPA_NO_LOAD

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_FREEZE << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

#### 4.29.2.5 #define EHRPWM_COMPB_LOAD_COUNT_EQUAL_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_TBCTRPRD << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.29.2.6 #define EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_TBCTRZERO << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.29.2.7 #define EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_ZEROORPRD << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.29.2.8 #define EHRPWM_COMPB_NO_LOAD

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_FREEZE << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.29.2.9 #define EHRPWM_COUNT_DOWN

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_DOWN << \
                            EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.29.2.10 #define EHRPWM_COUNT_STOP

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_STOPFREEZE << \
                            EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.29.2.11 #define EHRPWM_COUNT_UP

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_UP << \
                                EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.29.2.12 #define EHRPWM_COUNT_UP_DOWN

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_UPDOWN << \
                            EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.29.2.13 #define EHRPWM_SYNCOUT_COUNTER_EQUAL_COMPAREB

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_TBCTRCMPB << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

#### 4.29.2.14 #define EHRPWM_SYNCOUT_COUNTER_EQUAL_ZERO

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_TBCTRZERO << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

#### 4.29.2.15 #define EHRPWM_SYNCOUT_DISABLE

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_DISABLE << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

#### 4.29.2.16 #define EHRPWM_SYNCOUT_SYNCIN

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_EPWMXSYNCI << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

### 4.29.3 Function Documentation

#### 4.29.3.1 void EHRPWMAQContSWForceOnA ( unsigned int *baseAddr,* unsigned int *forceVal,* unsigned int *activeRegReloadMode* )

This API forces a value continuously on A. The output can be forced to low or high.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *forceVal* | Value to be forced |
| *activeReg↩ ReloadMode* | Shadow to active reg load trigger |

**Returns**

None

#### 4.29.3.2 void EHRPWMAQContSWForceOnB ( unsigned int *baseAddr,* unsigned int *forceVal,* unsigned int *activeRegReloadMode* )

This API forces a value continuously on B. The output can be forced to low or high.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *forceVal* | Value to be forced |
| *activeReg↩ ReloadMode* | Shadow to active reg load trigger |

**Returns**

None

#### 4.29.3.3 void EHRPWMChopperDisable ( unsigned int *baseAddr* )

This API disables the PWM chopper sub-module. This will cause the chopper module to be by-passed.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.4 void EHRPWMChopperEnable ( unsigned int *baseAddr* )

This API enables the PWM chopper sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.5 void EHRPWMClockDisable ( unsigned int *baseAdd* )

This functions enables clock for EHRPWM module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.29.3.6 unsigned int EHRPWMClockDisableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is disabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is disabled. return's '0' if clocked is not disabled.

#### 4.29.3.7 void EHRPWMClockEnable ( unsigned int *baseAdd* )

This functions enables clock for EHRPWM module in PWMSS subsystem.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

None.

#### 4.29.3.8 unsigned int EHRPWMClockEnableStatusGet ( unsigned int *baseAdd* )

This functions determines whether clock is enabled or not.

**Parameters**

| | |
|---|---|
| *baseAdd* | It is the Memory address of the PWMSS instance used. |

**Returns**

return's '1' if clocked is enabled. return's '0' if clocked is not enabled.

#### 4.29.3.9 void EHRPWMConfigHR ( unsigned int *baseAddr,* unsigned int *ctrlMode,* unsigned int *MEPCtrlEdge* )

This API configures control mode and edge mode. In also enables the HR sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *ctrlMode* | phase control or duty control |

| | |
|---|---|
| *MEPCtrlEdge* | Edge on which MEP to be applied (raising, falling, both) |

**Returns**

None

#### 4.29.3.10 void EHRPWMConfigureAQActionOnA ( unsigned int *baseAddr,* unsigned int *zero,* unsigned int *period,* unsigned int *CAUp,* unsigned int *CADown,* unsigned int *CBUp,* unsigned int *CBDown,* unsigned int *SWForced* )

This API configures the action to be taken on A by the Action qualifier module upon receiving the events. This will determine the output waveform.

**Parameters**

| | |
|---|---|
| *zero* | Action to be taken when CTR = 0 |
| *period* | Action to be taken when CTR = PRD |
| *CAUp* | Action to be taken when CTR = CAUp |
| *CADown* | Action to be taken when CTR = CADown |
| *CBUp* | Action to be taken when CTR = CBUp |
| *CBDown* | Action to be taken when CTR = CBDown |
| *SWForced* | Action to be taken when SW forced event has been generated<br>`Possible values for the actions are`<br>`- EHRPWM_XXXX_XXXX_DONOTHING \n`<br>`- EHRPWM_XXXX_XXXX_CLEAR \n`<br>`- EHRPWM_XXXX_XXXX_SET \n`<br>`- EHRPWM_XXXX_XXXX_TOGGLE \n` |

**Returns**

None

#### 4.29.3.11 void EHRPWMConfigureAQActionOnB ( unsigned int *baseAddr,* unsigned int *zero,* unsigned int *period,* unsigned int *CAUp,* unsigned int *CADown,* unsigned int *CBUp,* unsigned int *CBDown,* unsigned int *SWForced* )

his API configures the action to be taken on B by the Action qualifier module upon receiving the events. This will determine the output waveform.

**Parameters**

| | |
|---|---|
| *zero* | Action to be taken when CTR = 0 |
| *period* | Action to be taken when CTR = PRD |
| *CAUp* | Action to be taken when CTR = CAUp |
| *CADown* | Action to be taken when CTR = CADown |
| *CBUp* | Action to be taken when CTR = CBUp |
| *CBDown* | Action to be taken when CTR = CBDown |
| *SWForced* | Action to be taken when SW forced event has been generated<br>`Possible values for the actions are`<br>`- EHRPWM_XXXX_XXXX_DONOTHING \n`<br>`- EHRPWM_XXXX_XXXX_CLEAR \n`<br>`- EHRPWM_XXXX_XXXX_SET \n`<br>`- EHRPWM_XXXX_XXXX_TOGGLE \n` |

**Returns**

None

#### 4.29.3.12 void EHRPWMConfigureChopperDuty ( unsigned int *baseAddr,* unsigned int *dutyCycle* )

This API configures the chopper duty cyce. In Chopper sub-module the PWM signal is modulated with a carrier signal. Th duty cycle of the carrier signal is configured with this API.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *dutyCycle* | Duty cycle of the chopping carrier. Possible values are :<br>• EHRPWM_DUTY_12_5_PER<br>• EHRPWM_DUTY_25_PER<br>• EHRPWM_DUTY_37_5_PER<br>• EHRPWM_DUTY_50_PER<br>• EHRPWM_DUTY_62_5_PER<br>• EHRPWM_DUTY_75_PER<br>• EHRPWM_DUTY_87_5_PER |

**Returns**

None

#### 4.29.3.13 void EHRPWMConfigureChopperFreq ( unsigned int *baseAddr,* unsigned int *freqDiv* )

This API configures the chopper frequency. In chopper sub-module the PWM signal is modulated with a carrier signal. The frequency of the carrier signal is configured with this API.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *freqDiv* | Frequency divider |

**Returns**

None

#### 4.29.3.14 void EHRPWMConfigureChopperOSPW ( unsigned int *baseAddr,* unsigned int *OSPWCycles* )

This API configures one shot pulse width. The chopper module is useful in switching operations for pulse transformers. The one-shot block provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *OSPWCycles* | Number of clocks the OSPW to be ON. |

**Returns**

None

#### 4.29.3.15 void EHRPWMDBConfigureFED ( unsigned int *baseAddr,* unsigned int *fallingEdgeDelay* )

This API sets the Falling edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *fallingEdgeDelay* | Falling Edge Delay |

**Returns**

None

#### 4.29.3.16 void EHRPWMDBConfigureRED ( unsigned int *baseAddr,* unsigned int *raisingEdgeDelay* )

This API sets the raising edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *raisingEdge↩ Delay* | Raising Edge Delay |

**Returns**

None

#### 4.29.3.17 void EHRPWMDBOutput ( unsigned int *baseAddr,* unsigned int *DBgenOpMode* )

This API selects output mode. This allows to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenOpMode* | Output mode. The possible values can be :<br>• EHRPWM_DBCTL_OUT_MODE_BYPASS<br>• EHRPWM_DBCTL_OUT_MODE_NOREDBFED<br>• EHRPWM_DBCTL_OUT_MODE_AREDNOFED<br>• EHRPWM_DBCTL_OUT_MODE_AREDBFED |

**Returns**

None

#### 4.29.3.18 void EHRPWMDBPolaritySelect ( unsigned int *baseAddr,* unsigned int *DBgenPol* )

This API selects the polarity. This allows to selectively invert one of the delayed signals before it is sent out of the dead-band sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenSource* | Polarity. The possible values can be :<br>• HRPWM_DBCTL_POLSEL_ACTIVEHIGH<br>• EHRPWM_DBCTL_POLSEL_ALC<br>• EHRPWM_DBCTL_POLSEL_AHC<br>• EHRPWM_DBCTL_POLSEL_ACTIVELOW |

**Returns**

None

#### 4.29.3.19 void EHRPWMDBSourceSelect ( unsigned int *baseAddr,* unsigned int *DBgenSource* )

This API selects the source for delay blocks in dead band sub-module. The Dead band generator has two sub-modules, one for raising edge delay and the other for falling edge delay. This can be configured when a delay is need between two signals during signal change. The dead band generator is usefull in full-inverters.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenSource* | Source selection. The possible values can be<br>• EHRPWM_DBCTL_IN_MODE_AREDAFED<br>• EHRPWM_DBCTL_IN_MODE_BREDAFED<br>• EHRPWM_DBCTL_IN_MODE_AREDBFED<br>• EHRPWM_DBCTL_IN_MODE_BREDBFED |

**Returns**

None

#### 4.29.3.20 bool EHRPWMETEventCount ( unsigned int *baseAddr* )

This API returns the number of events occured.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

eventCount number of events occured

#### 4.29.3.21 void EHRPWMETIntClear ( unsigned int *baseAddr* )

This API clears the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.22 void EHRPWMETIntDisable ( unsigned int *baseAddr* )

This API disables the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.23 void EHRPWMETIntEnable ( unsigned int *baseAddr* )

This API enables the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.24 void EHRPWMETIntPrescale ( unsigned int *baseAddr,* unsigned int *prescale* )

This API prescales the event on which interrupt is to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *prescale* | prescalar value |

**Returns**

None

#### 4.29.3.25 void EHRPWMETIntSourceSelect ( unsigned int *baseAddr,* unsigned int *selectInt* )

This API selects the interrupt source.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *selectInt* | Event which triggers interrupt. The possible source can be,<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUZERO<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUPRD<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPAINC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPADEC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBINC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBDEC |

**Returns**

None

#### 4.29.3.26 bool EHRPWMETIntStatus ( unsigned int *baseAddr* )

This API returns the interrupt status.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

status Status of the interrupt

#### 4.29.3.27 void EHRPWMETIntSWForce ( unsigned int *baseAddr* )

This API forces interrupt to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.28 void EHRPWMHRDisable ( unsigned int *baseAddr* )

This API disables the HR sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.29 bool EHRPWMLoadCMPA ( unsigned int *baseAddr,* unsigned int *CMPAVal,* bool *enableShadowWrite,* unsigned int *ShadowToActiveLoadTrigger,* bool *OverwriteShadowFull* )

This API loads the CMPA value. When CMPA value equals the counter value, then an event is generated both in the up direction and down direction.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPAVal* | CMPA value to be loaded. |
| *enableShadow↩ Write* | Enable write to shadow register. |

| | |
|---|---|
| *ShadowToActiveLoadTrigger* | Shadow to active register load trigger. |
| *OverwriteShadowFull* | Overwrite even if previous value is not loaded to active register. |

**Returns**

bool Flag indicates whether the CMPA value is written or not.

#### 4.29.3.30 void EHRPWMLoadCMPAHR ( unsigned int *baseAddr,* unsigned int *CMPAHRVal,* unsigned int *ShadowToActiveLoadTrigger* )

This API loads CMPAHR value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPAHRVal* | CMPAHR value to he loaded |
| *ShadowToActiveLoadTrigger* | Condition when the active reg to be loaded from shadow register. |

**Returns**

None

#### 4.29.3.31 bool EHRPWMLoadCMPB ( unsigned int *baseAddr,* unsigned int *CMPBVal,* bool *enableShadowWrite,* unsigned int *ShadowToActiveLoadTrigger,* bool *OverwriteShadowFull* )

This API loads the CMPB value. When CMPB value equals the counter value, then an event is generated both in the up direction and down direction.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPBVal* | CMPB value to be loaded. |
| *enableShadowWrite* | Enable write to shadow register. |
| *ShadowToActiveLoadTrigger* | Shadow to active register load trigger. |
| *OverwriteShadowFull* | Overwrite even if previous value is not loaded to active register. |

**Returns**

bool Flag indicates whether the CMPB value is written or not.

#### 4.29.3.32 void EHRPWMLoadTBPHSHR ( unsigned int *baseAddr,* unsigned int *TBPHSHRVal* )

This API loads the HR PHS value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *TBPHSHRVal* | TB PHS HR value |

**Returns**

None

#### 4.29.3.33 void EHRPWMPWMOpFreqSet ( unsigned int *baseAddr,* unsigned int *tbClk,* unsigned int *pwmFreq,* unsigned int *counterDir,* bool *enableShadowWrite* )

This API configures the PWM Frequency/Period. The period count determines the period of the final output waveform. For the given period count, in the case of UP and DOWN counter the count value will be loaded as is. In the case of UP_DOWN counter the count is halfed.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbClk* | Timebase clock. |
| *pwmFreq* | Frequency of the PWM Output. If the counter direction is up-down this value has to be halfed, so that the period of the final output is equal to pwmFreq. |
| *counterDir* | Direction of the counter(up, down, up-down) |
| *enableShadowWrite* | Whether write to Period register is to be shadowed |

**Returns**

None.

#### 4.29.3.34 unsigned int EHRPWMReadTBCount ( unsigned int *baseAddr* )

This API gets the TB counter current value. The count operation is not affected by the read.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

tbCount Current Timebase count value.

#### 4.29.3.35 void EHRPWMSWForceA ( unsigned int *baseAddr* )

This API triggers the SW forced event on A. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.36 void EHRPWMSWForceB ( unsigned int *baseAddr* )

This API triggers the SW forced event on B. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.29.3.37 void EHRPWMSyncOutModeSet ( unsigned int *baseAddr,* unsigned int *syncOutMode* )

This API selects the output sync source. It determines on which of the following event sync-out has to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *syncOutMode* | Sync out mode. Possible values are,<br>• EHRPWM_SYNCOUT_SYNCIN<br>• EHRPWM_SYNCOUT_COUNTER_EQUAL_ZERO<br>• EHRPWM_SYNCOUT_COUNTER_EQUAL_COMPAREB<br>• EHRPWM_SYNCOUT_DISABLE |

**Returns**

None.

#### 4.29.3.38 void EHRPWMTBClearStatus ( unsigned int *baseAddr,* unsigned int *tbStatusMask* )

This API clears the TB status bits indicated by the tbStatusMask parameter.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbStatusMask* | Indicates which status bit need to be cleared.<br>• EHRPWM_TBSTS_CTRMAX<br>• EHRPWM_TBSTS_SYNCI<br>• EHRPWM_TBSTS_CTRDIR |

**Returns**

None

#### 4.29.3.39 void EHRPWMTBEmulationModeSet ( unsigned int *baseAddr,* unsigned int *mode* )

This API configures emulation mode. This setting determines the behaviour of Timebase during emulation (debugging).

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *mode* | Emulation mode. Possible values are,<br>• EHRPWM_STOP_AFTER_NEXT_TB_INCREMENT<br>• EHRPWM_STOP_AFTER_A_COMPLETE_CYCLE<br>• EHRPWM_FREE_RUN |

**Returns**

None.

#### 4.29.3.40 unsigned int EHRPWMTBStatusGet ( unsigned int *baseAddr,* unsigned int *tbStatusMask* )

This API gets the TB status as indicated by the tbStatusMask parameter.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbStatusMask* | Indicates which status is needed.<br>• EHRPWM_TBSTS_CTRMAX<br>– whether the counter has reached the max value<br>• EHRPWM_TBSTS_SYNCI<br>– indicates external sync event has occured<br>• EHRPWM_TBSTS_CTRDIR - gives the counter direction |

**Returns**

tbStatus Requested status is returned. The user need to extract the appropriate bits by shifting.

#### 4.29.3.41 void EHRPWMTimebaseClkConfig ( unsigned int *baseAddr,* unsigned int *tbClk,* unsigned int *moduleClk* )

This API configures the clock divider of the Time base module. The clock divider can be calculated using the equation TBCLK = SYSCLKOUT/(HSPCLKDIV LKDIV)

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbClk* | Timebase clock to be generated. |
| *moduleClk* | Input clock of the PWM module (sysclk2) |

**Returns**

None.

#### 4.29.3.42 void EHRPWMTimebaseSyncDisable ( unsigned int *baseAddr* )

This API disables the synchronization. Even if sync-in event occurs the count value will not be reloaded.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None.

#### 4.29.3.43 void EHRPWMTimebaseSyncEnable ( unsigned int *baseAddr,* unsigned int *tbPhsValue,* unsigned int *phsCountDir* )

This API enables the synchronization. When a sync-in event is generated the couter is reloaded with the new value. After sync the counter will use the new value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbPhsValue* | Phase value to be reloaded after sync |
| *phsCountDir* | Count direction after sync. Possible values are<br>• EHRPWM_COUNT_DOWN_AFTER_SYNC<br>• EHRPWM_COUNT_UP_AFTER_SYNC |

**Returns**

None.

#### 4.29.3.44 void EHRPWMTriggerSWSync ( unsigned int *baseAddr* )

This API generates sw forced sync pulse. This API can be used for testing. When this API is called sync-in will be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None.

#### 4.29.3.45 void EHRPWMTZFlagClear ( unsigned int *baseAddr,* unsigned int *flagToClear* )

This API clears the flag.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *flagToClear* | Status to be cleared. The possible values can be,<br>• EHRPWM_TZCLR_OST<br>• EHRPWM_TZCLR_CBC<br>• EHRPWM_TZCLR_INT |

**Returns**

None

#### 4.29.3.46 unsigned int EHRPWMTZFlagGet ( unsigned int *baseAddr,* unsigned int *flagToRead* )

This API returns the flag status requested.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *flagToRead* | status to be read. The possible values can be,<br>• EHRPWM_TZCLR_OST<br>• EHRPWM_TZCLR_CBC<br>• EHRPWM_TZCLR_INT |

**Returns**

None

#### 4.29.3.47 void EHRPWMTZForceAOnTrip ( unsigned int *baseAddr,* unsigned int *opValue* )

This API configures the o/p on A when a trip event is recognized. The output can be set to high or low or high impedence.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *opValue* | o/p state to be configured |

**Returns**

None

#### 4.29.3.48 void EHRPWMTZForceBOnTrip ( unsigned int *baseAddr,* unsigned int *opValue* )

This API configures the o/p on B when a trip event is recognised. The output can be set to high or low or high impedence.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *opValue* | o/p state to be configured |

**Returns**

None

#### 4.29.3.49 void EHRPWMTZIntDisable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API disables the trip interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | disable OST or CBC |

**Returns**

None

#### 4.29.3.50 void EHRPWMTZIntEnable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables the trip interrupt. When trip event occurs the sub-module can be configured to interrupt CPU.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

| | |
|---|---|
| *osht_CBC* | enable OST or CBC |

**Returns**

None

#### 4.29.3.51 void EHRPWMTZSWFrcEvent ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables to generate SW forced trip.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | generate OST or CBC trip |

**Returns**

None

#### 4.29.3.52 void EHRPWMTZTripEventDisable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API disable the trip event. The trip events will be ignored.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | Disable OST or CBC event |

**Returns**

None

#### 4.29.3.53 void EHRPWMTZTripEventEnable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables the trip event. The trip signals indicates external fault, and the ePWM outputs can be programmed to respond accordingly when faults occur.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | Enable OST or CBC event |

**Returns**

None

#### 4.29.3.54 void EHRPWMWriteTBCount ( unsigned int *baseAddr,* unsigned int *tbCount* )

This API loads the TB counter. The new value is taken immediately.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbCount* | Time base count value to be loaded. |

**Returns**

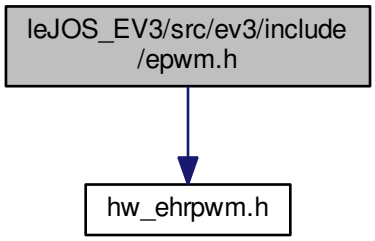
None.

## 4.30 leJOS_EV3/src/ev3/include/epwm.h File Reference

This file contains the Macros and API prototypes for ehrpwm driver.

```
#include "hw_ehrpwm.h"
```

Include dependency graph for epwm.h:




### Macros

- #define **EHRPWM_PRD_LOAD_SHADOW_MASK** EHRPWM_TBCTL_PRDLD
- #define **EHRPWM_COUNTER_MODE_MASK** EHRPWM_TBCTL_CTRMODE
- #define **EHRPWM_COUNT_UP**
- #define **EHRPWM_COUNT_DOWN**
- #define **EHRPWM_COUNT_UP_DOWN**
- #define **EHRPWM_COUNT_STOP**
- #define **EHRPWM_SYNC_ENABLE** EHRPWM_TBCTL_PHSEN
- #define **EHRPWM_SYNCOUT_MASK** EHRPWM_TBCTL_SYNCOSEL
- #define **EHRPWM_SYNCOUT_SYNCIN**
- #define **EHRPWM_SYNCOUT_COUNTER_EQUAL_ZERO**
- #define **EHRPWM_SYNCOUT_COUNTER_EQUAL_COMPAREB**
- #define **EHRPWM_SYNCOUT_DISABLE**
- #define **EHRPWM_SHADOW_WRITE_ENABLE** 0x0
- #define **EHRPWM_SHADOW_WRITE_DISABLE** 0x1
- #define **EHRPWM_STOP_AFTER_NEXT_TB_INCREMENT** (0x0 << EHRPWM_TBCTL_FREE_SOFT_↩ SHIFT)
- #define **EHRPWM_STOP_AFTER_A_COMPLETE_CYCLE** (0x1 << EHRPWM_TBCTL_FREE_SOFT_↩ SHIFT)
- #define **EHRPWM_FREE_RUN** (0x2 << EHRPWM_TBCTL_FREE_SOFT_SHIFT)
- #define **EHRPWM_TBCTL_CLKDIV_1** (0x0001u)
- #define **EHRPWM_TBCTL_CLKDIV_2** (0x0002u)

- #define **EHRPWM_TBCTL_CLKDIV_4** (0x0004u)
- #define **EHRPWM_TBCTL_CLKDIV_8** (0x0008u)
- #define **EHRPWM_TBCTL_CLKDIV_16** (0x0010u)
- #define **EHRPWM_TBCTL_CLKDIV_32** (0x0020u)
- #define **EHRPWM_TBCTL_CLKDIV_64** (0x0040u)
- #define **EHRPWM_TBCTL_CLKDIV_128** (0x0080u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_1** (0x0001u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_2** (0x0002u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_4** (0x0004u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_6** (0x0006u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_8** (0x0008u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_10** (0x000Au)
- #define **EHRPWM_TBCTL_HSPCLKDIV_12** (0x000Cu)
- #define **EHRPWM_TBCTL_HSPCLKDIV_14** (0x000Eu)
- #define **EHRPWM_COUNT_DOWN_AFTER_SYNC** 0x0
- #define **EHRPWM_COUNT_UP_AFTER_SYNC** 0x1
- #define **EHRPWM_SHADOW_A_EMPTY** (0x0 << EHRPWM_CMPCTL_SHDWAFULL_SHIFT)
- #define **EHRPWM_SHADOW_A_FULL** (EHRPWM_CMPCTL_SHDWAFULL)
- #define **EHRPWM_SHADOW_B_EMPTY** (0x0 << EHRPWM_CMPCTL_SHDWBFULL_SHIFT)
- #define **EHRPWM_SHADOW_B_FULL** (EHRPWM_CMPCTL_SHDWBFULL)
- #define **EHRPWM_CMPCTL_NOT_OVERWR_SH_FL** 0x0
- #define **EHRPWM_CMPCTL_OVERWR_SH_FL** 0x1
- #define **EHRPWM_COMPB_LOAD_MASK** EHRPWM_CMPCTL_LOADBMODE
- #define **EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO**
- #define **EHRPWM_COMPB_LOAD_COUNT_EQUAL_PERIOD**
- #define **EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD**
- #define **EHRPWM_COMPB_NO_LOAD**
- #define **EHRPWM_COMPA_LOAD_MASK** EHRPWM_CMPCTL_LOADAMODE
- #define **EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO**
- #define **EHRPWM_COMPA_LOAD_COUNT_EQUAL_PERIOD**
- #define **EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD**
- #define **EHRPWM_COMPA_NO_LOAD**
- #define **EHRPWM_CHP_DUTY_12_5_PER** EHRPWM_PCCTL_CHPDUTY_1DIV8
- #define **EHRPWM_CHP_DUTY_25_PER** EHRPWM_PCCTL_CHPDUTY_2DIV8
- #define **EHRPWM_CHP_DUTY_37_5_PER** EHRPWM_PCCTL_CHPDUTY_3DIV8
- #define **EHRPWM_CHP_DUTY_50_PER** EHRPWM_PCCTL_CHPDUTY_4DIV8
- #define **EHRPWM_CHP_DUTY_62_5_PER** EHRPWM_PCCTL_CHPDUTY_5DIV8
- #define **EHRPWM_CHP_DUTY_75_PER** EHRPWM_PCCTL_CHPDUTY_6DIV8
- #define **EHRPWM_CHP_DUTY_87_5_PER** EHRPWM_PCCTL_CHPDUTY_7DIV8
- #define **EHRPWM_TZ_ONESHOT** 0x0
- #define **EHRPWM_TZ_CYCLEBYCYCLE** 0x1
- #define **EHRPWM_TZ_ONESHOT_CLEAR** (EHRPWM_TZCLR_OST | EHRPWM_TZCLR_INT)
- #define **EHRPWM_TZ_CYCLEBYCYCLE_CLEAR** (EHRPWM_TZCLR_CBC | EHRPWM_TZCLR_INT)

## Typedefs

- typedef char **bool**

## Functions

- void EHRPWMTimebaseClkConfig (unsigned int baseAddr, unsigned int tbClk, unsigned int moduleClk)

  *This API configures the clock divider of the Time base module. The clock divider can be calculated using the equation TBCLK = SYSCLKOUT/(HSPCLKDIV LKDIV)*
- void EHRPWMPWMOpFreqSet (unsigned int baseAddr, unsigned int tbClk, unsigned int pwmFreq, unsigned int counterDir, bool enableShadowWrite)

  *This API configures the PWM Frequency/Period. The period count determines the period of the final output waveform. For the given period count, in the case of UP and DOWN counter the count value will be loaded as is. In the case of UP_DOWN counter the count is halfed.*
- void EHRPWMTBEmulationModeSet (unsigned int baseAddr, unsigned int mode)

  *This API configures emulation mode. This setting determines the behaviour of Timebase during emulation (debugging).*
- void EHRPWMTimebaseSyncEnable (unsigned int baseAddr, unsigned int tbPhsValue, unsigned int phs↩ CountDir)

  *This API enables the synchronization. When a sync-in event is generated the couter is reloaded with the new value. After sync the counter will use the new value.*
- void EHRPWMTimebaseSyncDisable (unsigned int baseAddr)

  *This API disables the synchronization. Even if sync-in event occurs the count value will not be reloaded.*
- void EHRPWMTriggerSWSync (unsigned int baseAddr)

  *This API generates sw forced sync pulse. This API can be used for testing. When this API is called sync-in will be generated.*
- void EHRPWMSyncOutModeSet (unsigned int baseAddr, unsigned int syncOutMode)

  *This API selects the output sync source. It determines on which of the following event sync-out has to be generated.*
- void EHRPWMWriteTBCount (unsigned int baseAddr, unsigned int tbCount)

  *This API loads the TB counter. The new value is taken immediately.*
- unsigned int EHRPWMReadTBCount (unsigned int baseAddr)

  *This API gets the TB counter current value. The count operation is not affected by the read.*
- unsigned int EHRPWMTBStatusGet (unsigned int baseAddr, unsigned int tbStatusMask)

  *This API gets the TB status as indicated by the tbStatusMask parameter.*
- void EHRPWMTBClearStatus (unsigned int baseAddr, unsigned int tbStatusMask)

  *This API clears the TB status bits indicated by the tbStatusMask parameter.*
- bool EHRPWMLoadCMPA (unsigned int baseAddr, unsigned int CMPAVal, bool enableShadowWrite, unsigned int ShadowToActiveLoadTrigger, bool OverwriteShadowFull)

  *This API loads the CMPA value. When CMPA value equals the counter value, then an event is generated both in the up direction and down direction.*
- bool EHRPWMLoadCMPB (unsigned int baseAddr, unsigned int CMPBVal, bool enableShadowWrite, unsigned int ShadowToActiveLoadTrigger, bool OverwriteShadowFull)

  *This API loads the CMPB value. When CMPB value equals the counter value, then an event is generated both in the up direction and down direction.*
- void EHRPWMConfigureAQActionOnA (unsigned int baseAddr, unsigned int zero, unsigned int period, unsigned int CAUp, unsigned int CADown, unsigned int CBUp, unsigned int CBDown, unsigned int SWForced)

  *This API configures the action to be taken on A by the Action qualifier module upon receiving the events. This will determine the output waveform.*
- void EHRPWMConfigureAQActionOnB (unsigned int baseAddr, unsigned int zero, unsigned int period, unsigned int CAUp, unsigned int CADown, unsigned int CBUp, unsigned int CBDown, unsigned int SWForced)

  *his API configures the action to be taken on B by the Action qualifier module upon receiving the events. This will determine the output waveform.*
- void EHRPWMSWForceA (unsigned int baseAddr)

  *This API triggers the SW forced event on A. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.*
- void EHRPWMSWForceB (unsigned int baseAddr)

  *This API triggers the SW forced event on B. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.*

- void EHRPWMAQContSWForceOnA (unsigned int baseAddr, unsigned int forceVal, unsigned int activeReg↩ ReloadMode)

  *This API forces a value continuously on A. The output can be forced to low or high.*
- void EHRPWMAQContSWForceOnB (unsigned int baseAddr, unsigned int forceVal, unsigned int activeReg↩ ReloadMode)

  *This API forces a value continuously on B. The output can be forced to low or high.*
- void EHRPWMDBSourceSelect (unsigned int baseAddr, unsigned int DBgenSource)

  *This API selects the source for delay blocks in dead band sub-module. The Dead band generator has two sub-modules, one for raising edge delay and the other for falling edge delay. This can be configured when a delay is need between two signals during signal change. The dead band generator is usefull in full-inverters.*
- void EHRPWMDBPolaritySelect (unsigned int baseAddr, unsigned int DBgenPol)

  *This API selects the polarity. This allows to selectively invert one of the delayed signals before it is sent out of the dead-band sub-module.*
- void EHRPWMDBOutput (unsigned int baseAddr, unsigned int DBgenOpMode)

  *This API selects output mode. This allows to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.*
- void EHRPWMDBConfigureRED (unsigned int baseAddr, unsigned int raisingEdgeDelay)

  *This API sets the raising edge delay.*
- void EHRPWMDBConfigureFED (unsigned int baseAddr, unsigned int fallingEdgeDelay)

  *This API sets the Falling edge delay.*
- void EHRPWMConfigureChopperDuty (unsigned int baseAddr, unsigned int dutyCycle)

  *This API configures the chopper duty cyce. In Chopper sub-module the PWM signal is modulated with a carrier signal. Th duty cycle of the carrier signal is configured with this API.*
- void EHRPWMConfigureChopperFreq (unsigned int baseAddr, unsigned int freqDiv)

  *This API configures the chopper frequency. In chopper sub-module the PWM signal is modulated with a carrier signal. The frequency of the carrier signal is configured with this API.*
- void EHRPWMConfigureChopperOSPW (unsigned int baseAddr, unsigned int OSPWCycles)

  *This API configures one shot pulse width. The chopper module is useful in switching operations for pulse transformers. The one-shot block provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on.*
- void EHRPWMChopperEnable (unsigned int baseAddr)

  *This API enables the PWM chopper sub-module.*
- void EHRPWMChopperDisable (unsigned int baseAddr)

  *This API disables the PWM chopper sub-module. This will cause the chopper module to be by-passed.*
- void EHRPWMTZTripEventEnable (unsigned int baseAddr, bool osht_CBC)

  *This API enables the trip event. The trip signals indicates external fault, and the ePWM outputs can be programmed to respond accordingly when faults occur.*
- void EHRPWMTZTripEventDisable (unsigned int baseAddr, bool osht_CBC)

  *This API disable the trip event. The trip events will be ignored.*
- void EHRPWMTZForceAOnTrip (unsigned int baseAddr, unsigned int opValue)

  *This API configures the o/p on A when a trip event is recognized. The output can be set to high or low or high impedence.*
- void EHRPWMTZForceBOnTrip (unsigned int baseAddr, unsigned int opValue)

  *This API configures the o/p on B when a trip event is recognised. The output can be set to high or low or high impedence.*
- void EHRPWMTZIntEnable (unsigned int baseAddr, bool osht_CBC)

  *This API enables the trip interrupt. When trip event occurs the sub-module can be configured to interrupt CPU.*
- void EHRPWMTZIntDisable (unsigned int baseAddr, bool osht_CBC)

  *This API disables the trip interrupt.*
- unsigned int EHRPWMTZFlagGet (unsigned int baseAddr, unsigned int flagToRead)

  *This API returns the flag status requested.*
- void EHRPWMTZFlagClear (unsigned int baseAddr, unsigned int flagToClear)

  *This API clears the flag.*

- void EHRPWMTZSWFrcEvent (unsigned int baseAddr, bool osht_CBC)

  *This API enables to generate SW forced trip.*
- void EHRPWMETIntDisable (unsigned int baseAddr)

  *This API disables the interrupt.*
- void EHRPWMETIntEnable (unsigned int baseAddr)

  *This API enables the interrupt.*
- void EHRPWMETIntSourceSelect (unsigned int baseAddr, unsigned int selectInt)

  *This API selects the interrupt source.*
- void EHRPWMETIntPrescale (unsigned int baseAddr, unsigned int prescale)

  *This API prescales the event on which interrupt is to be generated.*
- bool EHRPWMETEventCount (unsigned int baseAddr)

  *This API returns the number of events occured.*
- bool EHRPWMETIntStatus (unsigned int baseAddr)

  *This API returns the interrupt status.*
- void EHRPWMETIntClear (unsigned int baseAddr)

  *This API clears the interrupt.*
- void EHRPWMETIntSWForce (unsigned int baseAddr)

  *This API forces interrupt to be generated.*
- void EHRPWMLoadTBPHSHR (unsigned int baseAddr, unsigned int TBPHSHRVal)

  *This API loads the HR PHS value.*
- void EHRPWMLoadCMPAHR (unsigned int baseAddr, unsigned int CMPAHRVal, unsigned int ShadowTo↩ ActiveLoadTrigger)

  *This API loads CMPAHR value.*
- void EHRPWMConfigHR (unsigned int baseAddr, unsigned int ctrlMode, unsigned int MEPCtrlEdge)

  *This API configures control mode and edge mode. In also enables the HR sub-module.*
- void EHRPWMHRDisable (unsigned int baseAddr)

  *This API disables the HR sub-module.*

## 4.30.1 Detailed Description

This file contains the Macros and API prototypes for ehrpwm driver.

==========

## 4.30.2 Macro Definition Documentation

### 4.30.2.1 #define EHRPWM_COMPA_LOAD_COUNT_EQUAL_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_TBCTRPRD << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

### 4.30.2.2 #define EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_TBCTRZERO << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

#### 4.30.2.3 #define EHRPWM_COMPA_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_ZEROORPRD << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

#### 4.30.2.4 #define EHRPWM_COMPA_NO_LOAD

**Value:**

```
(EHRPWM_CMPCTL_LOADAMODE_FREEZE << \
                            EHRPWM_CMPCTL_LOADAMODE_SHIFT)
```

#### 4.30.2.5 #define EHRPWM_COMPB_LOAD_COUNT_EQUAL_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_TBCTRPRD << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.30.2.6 #define EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_TBCTRZERO << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.30.2.7 #define EHRPWM_COMPB_LOAD_COUNT_EQUAL_ZERO_OR_PERIOD

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_ZEROORPRD << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.30.2.8 #define EHRPWM_COMPB_NO_LOAD

**Value:**

```
(EHRPWM_CMPCTL_LOADBMODE_FREEZE << \
                            EHRPWM_CMPCTL_LOADBMODE_SHIFT)
```

#### 4.30.2.9 #define EHRPWM_COUNT_DOWN

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_DOWN << \
                            EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.30.2.10 #define EHRPWM_COUNT_STOP

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_STOPFREEZE << \
                            EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.30.2.11 #define EHRPWM_COUNT_UP

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_UP << \
                                EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.30.2.12 #define EHRPWM_COUNT_UP_DOWN

**Value:**

```
(EHRPWM_TBCTL_CTRMODE_UPDOWN << \
                            EHRPWM_TBCTL_CTRMODE_SHIFT)
```

#### 4.30.2.13 #define EHRPWM_SYNCOUT_COUNTER_EQUAL_COMPAREB

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_TBCTRCMPB << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

#### 4.30.2.14 #define EHRPWM_SYNCOUT_COUNTER_EQUAL_ZERO

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_TBCTRZERO << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

#### 4.30.2.15 #define EHRPWM_SYNCOUT_DISABLE

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_DISABLE << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

#### 4.30.2.16 #define EHRPWM_SYNCOUT_SYNCIN

**Value:**

```
(EHRPWM_TBCTL_SYNCOSEL_EPWMXSYNCI << \
                            EHRPWM_TBCTL_SYNCOSEL_SHIFT)
```

### 4.30.3 Function Documentation

#### 4.30.3.1 void EHRPWMAQContSWForceOnA ( unsigned int *baseAddr,* unsigned int *forceVal,* unsigned int *activeRegReloadMode* )

This API forces a value continuously on A. The output can be forced to low or high.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *forceVal* | Value to be forced |
| *activeReg↩ ReloadMode* | Shadow to active reg load trigger |

**Returns**

None

#### 4.30.3.2 void EHRPWMAQContSWForceOnB ( unsigned int *baseAddr,* unsigned int *forceVal,* unsigned int *activeRegReloadMode* )

This API forces a value continuously on B. The output can be forced to low or high.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *forceVal* | Value to be forced |
| *activeReg↩ ReloadMode* | Shadow to active reg load trigger |

**Returns**

None

#### 4.30.3.3 void EHRPWMChopperDisable ( unsigned int *baseAddr* )

This API disables the PWM chopper sub-module. This will cause the chopper module to be by-passed.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.4 void EHRPWMChopperEnable ( unsigned int *baseAddr* )

This API enables the PWM chopper sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.5 void EHRPWMConfigHR ( unsigned int *baseAddr,* unsigned int *ctrlMode,* unsigned int *MEPCtrlEdge* )

This API configures control mode and edge mode. In also enables the HR sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *ctrlMode* | phase control or duty control |
| *MEPCtrlEdge* | Edge on which MEP to be applied (raising, falling, both) |

**Returns**

None

#### 4.30.3.6 void EHRPWMConfigureAQActionOnA ( unsigned int *baseAddr,* unsigned int *zero,* unsigned int *period,* unsigned int *CAUp,* unsigned int *CADown,* unsigned int *CBUp,* unsigned int *CBDown,* unsigned int *SWForced* )

This API configures the action to be taken on A by the Action qualifier module upon receiving the events. This will determine the output waveform.

**Parameters**

| | |
|---|---|
| *zero* | Action to be taken when CTR = 0 |
| *period* | Action to be taken when CTR = PRD |
| *CAUp* | Action to be taken when CTR = CAUp |
| *CADown* | Action to be taken when CTR = CADown |
| *CBUp* | Action to be taken when CTR = CBUp |
| *CBDown* | Action to be taken when CTR = CBDown |
| *SWForced* | Action to be taken when SW forced event has been generated<br>Possible values for the actions are<br>– EHRPWM_XXXX_XXXX_DONOTHING \n<br>– EHRPWM_XXXX_XXXX_CLEAR \n<br>– EHRPWM_XXXX_XXXX_SET \n<br>– EHRPWM_XXXX_XXXX_TOGGLE \n |

**Returns**

None

#### 4.30.3.7 void EHRPWMConfigureAQActionOnB ( unsigned int *baseAddr,* unsigned int *zero,* unsigned int *period,* unsigned int *CAUp,* unsigned int *CADown,* unsigned int *CBUp,* unsigned int *CBDown,* unsigned int *SWForced* )

his API configures the action to be taken on B by the Action qualifier module upon receiving the events. This will determine the output waveform.

**Parameters**

| | |
|---|---|
| *zero* | Action to be taken when CTR = 0 |
| *period* | Action to be taken when CTR = PRD |
| *CAUp* | Action to be taken when CTR = CAUp |
| *CADown* | Action to be taken when CTR = CADown |
| *CBUp* | Action to be taken when CTR = CBUp |
| *CBDown* | Action to be taken when CTR = CBDown |
| *SWForced* | Action to be taken when SW forced event has been generated<br>Possible values for the actions are<br>– EHRPWM_XXXX_XXXX_DONOTHING \n<br>– EHRPWM_XXXX_XXXX_CLEAR \n<br>– EHRPWM_XXXX_XXXX_SET \n<br>– EHRPWM_XXXX_XXXX_TOGGLE \n |

**Returns**

None

#### 4.30.3.8 void EHRPWMConfigureChopperDuty ( unsigned int *baseAddr,* unsigned int *dutyCycle* )

This API configures the chopper duty cyce. In Chopper sub-module the PWM signal is modulated with a carrier signal. Th duty cycle of the carrier signal is configured with this API.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *dutyCycle* | Duty cycle of the chopping carrier. Possible values are :<br>• EHRPWM_DUTY_12_5_PER<br>• EHRPWM_DUTY_25_PER<br>• EHRPWM_DUTY_37_5_PER<br>• EHRPWM_DUTY_50_PER<br>• EHRPWM_DUTY_62_5_PER<br>• EHRPWM_DUTY_75_PER<br>• EHRPWM_DUTY_87_5_PER |

**Returns**

None

#### 4.30.3.9 void EHRPWMConfigureChopperFreq ( unsigned int *baseAddr,* unsigned int *freqDiv* )

This API configures the chopper frequency. In chopper sub-module the PWM signal is modulated with a carrier signal. The frequency of the carrier signal is configured with this API.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *freqDiv* | Frequency divider |

**Returns**

None

#### 4.30.3.10 void EHRPWMConfigureChopperOSPW ( unsigned int *baseAddr,* unsigned int *OSPWCycles* )

This API configures one shot pulse width. The chopper module is useful in switching operations for pulse transformers. The one-shot block provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *OSPWCycles* | Number of clocks the OSPW to be ON. |

**Returns**

None

#### 4.30.3.11 void EHRPWMDBConfigureFED ( unsigned int *baseAddr,* unsigned int *fallingEdgeDelay* )

This API sets the Falling edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *fallingEdgeDelay* | Falling Edge Delay |

**Returns**

None

#### 4.30.3.12 void EHRPWMDBConfigureRED ( unsigned int *baseAddr,* unsigned int *raisingEdgeDelay* )

This API sets the raising edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *raisingEdge↩ Delay* | Raising Edge Delay |

**Returns**

None

#### 4.30.3.13 void EHRPWMDBOutput ( unsigned int *baseAddr,* unsigned int *DBgenOpMode* )

This API selects output mode. This allows to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenOpMode* | Output mode. The possible values can be :<br>• EHRPWM_DBCTL_OUT_MODE_BYPASS<br>• EHRPWM_DBCTL_OUT_MODE_NOREDBFED<br>• EHRPWM_DBCTL_OUT_MODE_AREDNOFED<br>• EHRPWM_DBCTL_OUT_MODE_AREDBFED |

**Returns**

None

#### 4.30.3.14 void EHRPWMDBPolaritySelect ( unsigned int *baseAddr,* unsigned int *DBgenPol* )

This API selects the polarity. This allows to selectively invert one of the delayed signals before it is sent out of the dead-band sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenSource* | Polarity. The possible values can be :<br>• HRPWM_DBCTL_POLSEL_ACTIVEHIGH<br>• EHRPWM_DBCTL_POLSEL_ALC<br>• EHRPWM_DBCTL_POLSEL_AHC<br>• EHRPWM_DBCTL_POLSEL_ACTIVELOW |

**Returns**

None

#### 4.30.3.15 void EHRPWMDBSourceSelect ( unsigned int *baseAddr,* unsigned int *DBgenSource* )

This API selects the source for delay blocks in dead band sub-module. The Dead band generator has two sub-modules, one for raising edge delay and the other for falling edge delay. This can be configured when a delay is need between two signals during signal change. The dead band generator is usefull in full-inverters.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *DBgenSource* | Source selection. The possible values can be<br>• EHRPWM_DBCTL_IN_MODE_AREDAFED<br>• EHRPWM_DBCTL_IN_MODE_BREDAFED<br>• EHRPWM_DBCTL_IN_MODE_AREDBFED<br>• EHRPWM_DBCTL_IN_MODE_BREDBFED |

**Returns**

None

#### 4.30.3.16 bool EHRPWMETEventCount ( unsigned int *baseAddr* )

This API returns the number of events occured.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

eventCount number of events occured

#### 4.30.3.17 void EHRPWMETIntClear ( unsigned int *baseAddr* )

This API clears the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.18 void EHRPWMETIntDisable ( unsigned int *baseAddr* )

This API disables the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.19 void EHRPWMETIntEnable ( unsigned int *baseAddr* )

This API enables the interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.20 void EHRPWMETIntPrescale ( unsigned int *baseAddr,* unsigned int *prescale* )

This API prescales the event on which interrupt is to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *prescale* | prescalar value |

**Returns**

None

#### 4.30.3.21 void EHRPWMETIntSourceSelect ( unsigned int *baseAddr,* unsigned int *selectInt* )

This API selects the interrupt source.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *selectInt* | Event which triggers interrupt. The possible source can be,<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUZERO<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUPRD<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPAINC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPADEC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBINC<br>• EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBDEC |

**Returns**

None

#### 4.30.3.22 bool EHRPWMETIntStatus ( unsigned int *baseAddr* )

This API returns the interrupt status.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

status Status of the interrupt

#### 4.30.3.23 void EHRPWMETIntSWForce ( unsigned int *baseAddr* )

This API forces interrupt to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.24 void EHRPWMHRDisable ( unsigned int *baseAddr* )

This API disables the HR sub-module.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.25 bool EHRPWMLoadCMPA ( unsigned int *baseAddr,* unsigned int *CMPAVal,* bool *enableShadowWrite,* unsigned int *ShadowToActiveLoadTrigger,* bool *OverwriteShadowFull* )

This API loads the CMPA value. When CMPA value equals the counter value, then an event is generated both in the up direction and down direction.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPAVal* | CMPA value to be loaded. |
| *enableShadow↩ Write* | Enable write to shadow register. |

| | |
|---|---|
| *ShadowToActiveLoadTrigger* | Shadow to active register load trigger. |
| *OverwriteShadowFull* | Overwrite even if previous value is not loaded to active register. |

**Returns**

bool Flag indicates whether the CMPA value is written or not.

#### 4.30.3.26 void EHRPWMLoadCMPAHR ( unsigned int *baseAddr,* unsigned int *CMPAHRVal,* unsigned int *ShadowToActiveLoadTrigger* )

This API loads CMPAHR value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPAHRVal* | CMPAHR value to he loaded |
| *ShadowToActiveLoadTrigger* | Condition when the active reg to be loaded from shadow register. |

**Returns**

None

#### 4.30.3.27 bool EHRPWMLoadCMPB ( unsigned int *baseAddr,* unsigned int *CMPBVal,* bool *enableShadowWrite,* unsigned int *ShadowToActiveLoadTrigger,* bool *OverwriteShadowFull* )

This API loads the CMPB value. When CMPB value equals the counter value, then an event is generated both in the up direction and down direction.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *CMPBVal* | CMPB value to be loaded. |
| *enableShadowWrite* | Enable write to shadow register. |
| *ShadowToActiveLoadTrigger* | Shadow to active register load trigger. |
| *OverwriteShadowFull* | Overwrite even if previous value is not loaded to active register. |

**Returns**

bool Flag indicates whether the CMPB value is written or not.

#### 4.30.3.28 void EHRPWMLoadTBPHSHR ( unsigned int *baseAddr,* unsigned int *TBPHSHRVal* )

This API loads the HR PHS value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *TBPHSHRVal* | TB PHS HR value |

**Returns**

None

#### 4.30.3.29 void EHRPWMPWMOpFreqSet ( unsigned int *baseAddr,* unsigned int *tbClk,* unsigned int *pwmFreq,* unsigned int *counterDir,* bool *enableShadowWrite* )

This API configures the PWM Frequency/Period. The period count determines the period of the final output waveform. For the given period count, in the case of UP and DOWN counter the count value will be loaded as is. In the case of UP_DOWN counter the count is halfed.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbClk* | Timebase clock. |
| *pwmFreq* | Frequency of the PWM Output. If the counter direction is up-down this value has to be halfed, so that the period of the final output is equal to pwmFreq. |
| *counterDir* | Direction of the counter(up, down, up-down) |
| *enableShadowWrite* | Whether write to Period register is to be shadowed |

**Returns**

None.

#### 4.30.3.30 unsigned int EHRPWMReadTBCount ( unsigned int *baseAddr* )

This API gets the TB counter current value. The count operation is not affected by the read.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

tbCount Current Timebase count value.

#### 4.30.3.31 void EHRPWMSWForceA ( unsigned int *baseAddr* )

This API triggers the SW forced event on A. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.32 void EHRPWMSWForceB ( unsigned int *baseAddr* )

This API triggers the SW forced event on B. This can be used for testing the AQ sub-module.Every call to this API will trigger a single event.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None

#### 4.30.3.33 void EHRPWMSyncOutModeSet ( unsigned int *baseAddr,* unsigned int *syncOutMode* )

This API selects the output sync source. It determines on which of the following event sync-out has to be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *syncOutMode* | Sync out mode. Possible values are,<br>• EHRPWM_SYNCOUT_SYNCIN<br>• EHRPWM_SYNCOUT_COUNTER_EQUAL_ZERO<br>• EHRPWM_SYNCOUT_COUNTER_EQUAL_COMPAREB<br>• EHRPWM_SYNCOUT_DISABLE |

**Returns**

None.

#### 4.30.3.34 void EHRPWMTBClearStatus ( unsigned int *baseAddr,* unsigned int *tbStatusMask* )

This API clears the TB status bits indicated by the tbStatusMask parameter.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbStatusMask* | Indicates which status bit need to be cleared.<br>• EHRPWM_TBSTS_CTRMAX<br>• EHRPWM_TBSTS_SYNCI<br>• EHRPWM_TBSTS_CTRDIR |

**Returns**

None

#### 4.30.3.35 void EHRPWMTBEmulationModeSet ( unsigned int *baseAddr,* unsigned int *mode* )

This API configures emulation mode. This setting determines the behaviour of Timebase during emulation (debugging).

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *mode* | Emulation mode. Possible values are,<br>• EHRPWM_STOP_AFTER_NEXT_TB_INCREMENT<br>• EHRPWM_STOP_AFTER_A_COMPLETE_CYCLE<br>• EHRPWM_FREE_RUN |

**Returns**

None.

#### 4.30.3.36 unsigned int EHRPWMTBStatusGet ( unsigned int *baseAddr,* unsigned int *tbStatusMask* )

This API gets the TB status as indicated by the tbStatusMask parameter.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbStatusMask* | Indicates which status is needed.<br>• EHRPWM_TBSTS_CTRMAX<br>– whether the counter has reached the max value<br>• EHRPWM_TBSTS_SYNCI<br>– indicates external sync event has occured<br>• EHRPWM_TBSTS_CTRDIR - gives the counter direction |

**Returns**

tbStatus Requested status is returned. The user need to extract the appropriate bits by shifting.

#### 4.30.3.37 void EHRPWMTimebaseClkConfig ( unsigned int *baseAddr,* unsigned int *tbClk,* unsigned int *moduleClk* )

This API configures the clock divider of the Time base module. The clock divider can be calculated using the equation TBCLK = SYSCLKOUT/(HSPCLKDIV LKDIV)

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbClk* | Timebase clock to be generated. |
| *moduleClk* | Input clock of the PWM module (sysclk2) |

**Returns**

None.

#### 4.30.3.38 void EHRPWMTimebaseSyncDisable ( unsigned int *baseAddr* )

This API disables the synchronization. Even if sync-in event occurs the count value will not be reloaded.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None.

#### 4.30.3.39 void EHRPWMTimebaseSyncEnable ( unsigned int *baseAddr,* unsigned int *tbPhsValue,* unsigned int *phsCountDir* )

This API enables the synchronization. When a sync-in event is generated the couter is reloaded with the new value. After sync the counter will use the new value.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbPhsValue* | Phase value to be reloaded after sync |
| *phsCountDir* | Count direction after sync. Possible values are<br>• EHRPWM_COUNT_DOWN_AFTER_SYNC<br>• EHRPWM_COUNT_UP_AFTER_SYNC |

**Returns**

None.

#### 4.30.3.40 void EHRPWMTriggerSWSync ( unsigned int *baseAddr* )

This API generates sw forced sync pulse. This API can be used for testing. When this API is called sync-in will be generated.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

**Returns**

None.

#### 4.30.3.41 void EHRPWMTZFlagClear ( unsigned int *baseAddr,* unsigned int *flagToClear* )

This API clears the flag.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *flagToClear* | Status to be cleared. The possible values can be,<br>• EHRPWM_TZCLR_OST<br>• EHRPWM_TZCLR_CBC<br>• EHRPWM_TZCLR_INT |

**Returns**

None

#### 4.30.3.42 unsigned int EHRPWMTZFlagGet ( unsigned int *baseAddr,* unsigned int *flagToRead* )

This API returns the flag status requested.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *flagToRead* | status to be read. The possible values can be,<br>• EHRPWM_TZCLR_OST<br>• EHRPWM_TZCLR_CBC<br>• EHRPWM_TZCLR_INT |

**Returns**

None

#### 4.30.3.43 void EHRPWMTZForceAOnTrip ( unsigned int *baseAddr,* unsigned int *opValue* )

This API configures the o/p on A when a trip event is recognized. The output can be set to high or low or high impedence.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *opValue* | o/p state to be configured |

**Returns**

None

#### 4.30.3.44 void EHRPWMTZForceBOnTrip ( unsigned int *baseAddr,* unsigned int *opValue* )

This API configures the o/p on B when a trip event is recognised. The output can be set to high or low or high impedence.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *opValue* | o/p state to be configured |

**Returns**

None

#### 4.30.3.45 void EHRPWMTZIntDisable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API disables the trip interrupt.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | disable OST or CBC |

**Returns**

None

#### 4.30.3.46 void EHRPWMTZIntEnable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables the trip interrupt. When trip event occurs the sub-module can be configured to interrupt CPU.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |

| | |
|---|---|
| *osht_CBC* | enable OST or CBC |

**Returns**

None

#### 4.30.3.47 void EHRPWMTZSWFrcEvent ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables to generate SW forced trip.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | generate OST or CBC trip |

**Returns**

None

#### 4.30.3.48 void EHRPWMTZTripEventDisable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API disable the trip event. The trip events will be ignored.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | Disable OST or CBC event |

**Returns**

None

#### 4.30.3.49 void EHRPWMTZTripEventEnable ( unsigned int *baseAddr,* bool *osht_CBC* )

This API enables the trip event. The trip signals indicates external fault, and the ePWM outputs can be programmed to respond accordingly when faults occur.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *osht_CBC* | Enable OST or CBC event |

**Returns**

None

#### 4.30.3.50 void EHRPWMWriteTBCount ( unsigned int *baseAddr,* unsigned int *tbCount* )

This API loads the TB counter. The new value is taken immediately.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the PWM Module Registers. |
| *tbCount* | Time base count value to be loaded. |

**Returns**

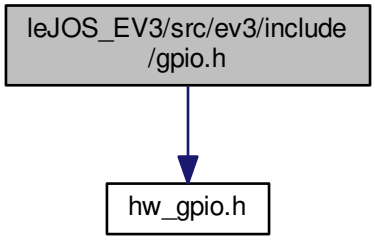
None.

## 4.31 leJOS_EV3/src/ev3/include/gpio.h File Reference

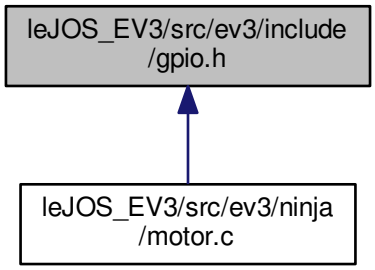
This file contains the function prototypes for the device abstraction layer for GPIO and some related macros.

```
#include "hw_gpio.h"
```

Include dependency graph for gpio.h:




This graph shows which files directly or indirectly include this file:




### Macros

- #define **GPIO_DIR_INPUT** 1
- #define **GPIO_DIR_OUTPUT** 0
- #define **GPIO_INT_TYPE_NOEDGE** 0

- #define **GPIO_INT_TYPE_FALLEDGE** 1
- #define **GPIO_INT_TYPE_RISEDGE** 2
- #define **GPIO_INT_TYPE_BOTHEDGE** 3
- #define **GPIO_INT_NOPEND** 0
- #define **GPIO_INT_PEND** 1
- #define **GPIO_PIN_LOW** 0
- #define **GPIO_PIN_HIGH** 1
- #define **GPIO_BANK_PIN_0** GPIO_DIR_DIR0
- #define **GPIO_BANK_PIN_1** GPIO_DIR_DIR1
- #define **GPIO_BANK_PIN_2** GPIO_DIR_DIR2
- #define **GPIO_BANK_PIN_3** GPIO_DIR_DIR3
- #define **GPIO_BANK_PIN_4** GPIO_DIR_DIR4
- #define **GPIO_BANK_PIN_5** GPIO_DIR_DIR5
- #define **GPIO_BANK_PIN_6** GPIO_DIR_DIR6
- #define **GPIO_BANK_PIN_7** GPIO_DIR_DIR7
- #define **GPIO_BANK_PIN_8** GPIO_DIR_DIR8
- #define **GPIO_BANK_PIN_9** GPIO_DIR_DIR9
- #define **GPIO_BANK_PIN_10** GPIO_DIR_DIR10
- #define **GPIO_BANK_PIN_11** GPIO_DIR_DIR11
- #define **GPIO_BANK_PIN_12** GPIO_DIR_DIR12
- #define **GPIO_BANK_PIN_13** GPIO_DIR_DIR13
- #define **GPIO_BANK_PIN_14** GPIO_DIR_DIR14
- #define **GPIO_BANK_PIN_15** GPIO_DIR_DIR15

## Functions

- void GPIODirModeSet (unsigned int baseAdd, unsigned int pinNumber, unsigned int pinDir)

  *This function configures the direction of a pin as input or output.*
- unsigned int GPIODirModeGet (unsigned int baseAdd, unsigned int pinNumber)

  *This function gets the direction of a pin which has been configured as an input or an output pin.*
- void GPIOPinWrite (unsigned int baseAdd, unsigned int pinNumber, unsigned int bitValue)

  *This function writes a logic 1 or a logic 0 to the specified pin.*
- int GPIOPinRead (unsigned int baseAdd, unsigned int pinNumber)

  *This function reads the value(logic level) of an input or an output pin.*
- void GPIOIntTypeSet (unsigned int baseAdd, unsigned int pinNumber, unsigned int intType)

  *This function configures the trigger level type for which an interrupt is required to occur.*
- unsigned int GPIOIntTypeGet (unsigned int baseAdd, unsigned int pinNumber)

  *This function reads the trigger level type being set for interrupts to be generated.*
- unsigned int GPIOPinIntStatus (unsigned int baseAdd, unsigned int pinNumber)

  *This function determines the status of interrupt on a specified pin.*
- void GPIOPinIntClear (unsigned int baseAdd, unsigned int pinNumber)

  *This function clears the interrupt status of the pin being accessed.*
- void GPIOBankIntEnable (unsigned int baseAdd, unsigned int bankNumber)

  *This function enables the interrupt generation capability for the bank of GPIO pins specified.*
- void GPIOBankIntDisable (unsigned int baseAdd, unsigned int bankNumber)

  *This function disables the interrupt generation capability for the bank of GPIO pins specified.*
- void GPIOBankPinsWrite (unsigned int baseAdd, unsigned int bankNumber, unsigned int setPins, unsigned int clrPins)

  *This function is used to collectively set and collectively clear the specified bits.*

### 4.31.1 Detailed Description

This file contains the function prototypes for the device abstraction layer for GPIO and some related macros.

### 4.31.2 Function Documentation

#### 4.31.2.1 void GPIOBankIntDisable ( unsigned int *baseAdd,* unsigned int *bankNumber* )

This function disables the interrupt generation capability for the bank of GPIO pins specified.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *bankNumber* | This is the bank for whose pins interrupt generation capability needs to be disabled. bank↩ Number is 0 for bank 0, 1 for bank 1 and so on. |

**Returns**

None.

#### 4.31.2.2 void GPIOBankIntEnable ( unsigned int *baseAdd,* unsigned int *bankNumber* )

This function enables the interrupt generation capability for the bank of GPIO pins specified.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *bankNumber* | This is the bank for whose pins interrupt generation capabiility needs to be enabled. bank↩ Number is 0 for bank 0, 1 for bank 1 and so on. |

**Returns**

None.

#### 4.31.2.3 void GPIOBankPinsWrite ( unsigned int *baseAdd,* unsigned int *bankNumber,* unsigned int *setPins,* unsigned int *clrPins* )

This function is used to collectively set and collectively clear the specified bits.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *bankNumber* | Numerical value of the bank whose pins are to be modified. |
| *setPins* | The bit-mask of the pins whose values have to be set. This could be the bitwise OR of the following macros: -> GPIO_BANK_PIN_n where n >= 0 and n <= 15. |
| *clrPins* | The bit-mask of the pins whose values have to be cleared. This could be the bitwise OR of the following macros: -> GPIO_BANK_PIN_n where n >= 0 and n <= 15. |

**Returns**

None.

**Note**

The pre-requisite to write to any pins is that the pins have to be configured as output pins.

#### 4.31.2.4 unsigned int GPIODirModeGet ( unsigned int *baseAdd,* unsigned int *pinNumber* )

This function gets the direction of a pin which has been configured as an input or an output pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns one of the following two values: 1> GPIO_DIR_INPUT, if the pin is configured as an input pin. 2> GPIO_DIR_OUTPUT, if the pin is configured as an output pin.

#### 4.31.2.5 void GPIODirModeSet ( unsigned int *baseAdd,* unsigned int *pinNumber,* unsigned int *pinDir* )

This function configures the direction of a pin as input or output.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |
| *pinDir* | The direction to be set for the pin. This can take the values: 1> GPIO_DIR_INPUT, for configuring the pin as input. 2> GPIO_DIR_OUTPUT, for configuring the pin as output. |

**Returns**

None.

**Note**

Here we write to the DIRn register. Writing a logic 1 configures the pin as input and writing logic 0 as output. By default, all the pins are set as input pins.

#### 4.31.2.6 unsigned int GPIOIntTypeGet ( unsigned int *baseAdd,* unsigned int *pinNumber* )

This function reads the trigger level type being set for interrupts to be generated.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin to be accessed. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns a value which indicates the type of trigger level type being set. One of the following values is returned. 1> GPIO_INT_TYPE_NOEDGE, indicating no interrupts will be generated over the corresponding pin. 2> GPIO_INT_TYPE_FALLEDGE, indicating a falling edge on the corresponding pin signifies an interrupt generation. 3> GPIO_INT_TYPE_RISEDGE, indicating a rising edge on the corresponding pin signifies an interrupt generation. 4> GPIO_INT_TYPE_BOTHEDGE, indicating both edges on the corresponding pin signifies an interrupt each being generated.

#### 4.31.2.7 void GPIOIntTypeSet ( unsigned int *baseAdd,* unsigned int *pinNumber,* unsigned int *intType* )

This function configures the trigger level type for which an interrupt is required to occur.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |
| *intType* | This specifies the trigger level type. This can take one of the following four values: 1> GPIO_INT_TYPE_NOEDGE, to not generate any interrupts. 2> GPIO_INT_TYPE_FALLEDGE, to generate an interrupt on the falling edge of a signal on that pin. 3> GPIO_INT_TYPE_RISEDGE, to generate an interrupt on the rising edge of a signal on that pin. 4> GPIO_INT_TYPE_BOTHEDGE, to generate interrupts on both rising and falling edges of a signal on that pin. |

**Returns**

None.

**Note**

Configuring the trigger level type for generating interrupts is not enough for the GPIO module to generate interrupts. The user should also enable the interrupt generation capability for the bank to which the pin belongs to. Use the function GPIOBankIntEnable() to do the same.

#### 4.31.2.8 void GPIOPinIntClear ( unsigned int *baseAdd*, unsigned int *pinNumber* )

This function clears the interrupt status of the pin being accessed.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin to be accessed. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

None.

#### 4.31.2.9 unsigned int GPIOPinIntStatus ( unsigned int *baseAdd*, unsigned int *pinNumber* )

This function determines the status of interrupt on a specified pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin to be accessed. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns a value which expresses the status of an interrupt raised over the specified pin. 1> GPIO_INT_NOPEND, if no interrupts are left to be serviced. 2> GPIO_INT_PEND, if the interrupt raised over that pin is yet to be cleared and serviced.

**Note**

If an interrupt over a pin is found to be pending, then the application can call GPIOPinIntClear() to clear the interrupt status.

#### 4.31.2.10 int GPIOPinRead ( unsigned int *baseAdd*, unsigned int *pinNumber* )

This function reads the value(logic level) of an input or an output pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |

**Returns**

This returns the value present on the specified pin. This returns one of the following values: 1> GPIO_PIN↩ _LOW, if the value on the pin is logic 0. 2> GPIO_PIN_HIGH, if the value on the pin is logic 1.

**Note**

Using this function, we can read the values of both input and output pins.

#### 4.31.2.11 void GPIOPinWrite ( unsigned int *baseAdd,* unsigned int *pinNumber,* unsigned int *bitValue* )

This function writes a logic 1 or a logic 0 to the specified pin.

**Parameters**

| | |
|---|---|
| *baseAdd* | The memory address of the GPIO instance being used. |
| *pinNumber* | The serial number of the GPIO pin. The 144 GPIO pins have serial numbers from 1 to 144. |
| *bitValue* | This signifies whether to write a logic 0 or logic 1 to the specified pin.This variable can take any of the following two values: 1> GPIO_PIN_LOW, which indicates to clear(logic 0) the bit. 2> GPIO_PIN_HIGH, which indicates to set(logic 1) the bit. |

**Returns**

None.

**Note**

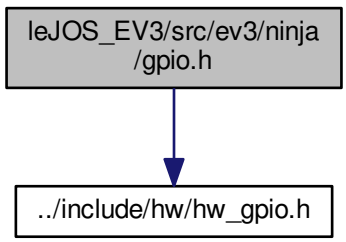
The pre-requisite to write to any pin is that the pin has to be configured as an output pin.

## 4.32 leJOS_EV3/src/ev3/ninja/gpio.h File Reference

This header declares function required to interact with GPIO pins of the SoC.

```
#include "../include/hw/hw_gpio.h"
```

Include dependency graph for gpio.h:



This graph shows which files directly or indirectly include this file:

## Macros

- #define SYSCFG0_BASE ((volatile void∗)0x01C14000)

  *The base address of the SYSCFG0 register.*
- #define SYSCFG1_BASE ((volatile void∗)0x01E2C000)

  *The base address of the SYSCFG1 register.*
- #define GPIO_BASE ((volatile void∗)0x01E26000)

  *The base address of the GPIO control registers.*
- #define GPIO_PIN(B, O) ((B) ∗ 0x10 + (O))

  *Calculate the number of a GPIO pin (0 to 143) based on its bank and number on that bank.*
- #define GPIO_BANK(N) (GPIO_BASE + 0x10 + (N >> 5) ∗ 0x28)

  *Get the base address of the registers responsible for controlling a specific GPIO pin based on the pin number.*
- #define GPIO_MASK(N) (1 << (N & 0x1F))

  *Get the mask required to manipulate the registers for the GPIO pin based on its pin number.*
- #define GPIO_DIR(N) ∗((volatile unsigned int∗)(GPIO_BANK(N) + 0x00))

  *A dereferenced pointer to the GPIO direction register for a pin based on the pin number.*
- #define GPIO_SET(N) ∗((volatile unsigned int∗)(GPIO_BANK(N) + 0x08))

  *A dereferenced pointer to the GPIO set register for a pin based on the pin number.*
- #define GPIO_CLR(N) ∗((volatile unsigned int∗)(GPIO_BANK(N) + 0x0C))

  *A dereferenced pointer to the GPIO clear register for a pin based on the pin number.*

## Functions

- void gpio_init_pin (unsigned int pin)

  *Initialize a GPIO pin with its GPIO functionality.*
- void gpio_init_outpin (unsigned int pin)

  *Initialize a GPIO pin with its GPIO functionality and set its direction to output.*
- void gpio_init_inpin (unsigned int pin)

  *Initialize a GPIO pin with its GPIO functionality and set its direction to input.*
- void gpio_set (unsigned int pin, unsigned int value)

  *Set the value of a GPIO pin which is configured as output.*
- unsigned int gpio_get (unsigned int pin)

  *Get the signal at a GPIO pin which is configured as input.*
- void **spi_init_pin** (unsigned int pin)
- void turn_off_brick (void)

  *Turn the EV3 off by setting GPIO pin 6 on bank 11 as an output pin.*

### 4.32.1 Detailed Description

This header declares function required to interact with GPIO pins of the SoC.

**Author**

Tobias Schießl, ev3ninja

### 4.32.2 Function Documentation

#### 4.32.2.1 unsigned int gpio_get ( unsigned int *pin* )

Get the signal at a GPIO pin which is configured as input.

If not already done, this function will set the pin as an input pin.

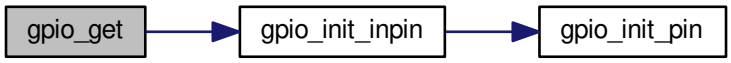
**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

The signal at the pin: 1 for high and 0 for low

Here is the call graph for this function:




#### 4.32.2.2 void gpio_init_inpin ( unsigned int *pin* )

Initialize a GPIO pin with its GPIO functionality and set its direction to input.

**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

none

Here is the call graph for this function:


#### 4.32.2.3 void gpio_init_outpin ( unsigned int *pin* )

Initialize a GPIO pin with its GPIO functionality and set its direction to output.

**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

none

Here is the call graph for this function:


#### 4.32.2.4 void gpio_init_pin ( unsigned int *pin* )

Initialize a GPIO pin with its GPIO functionality.

**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |

**Returns**

none

#### 4.32.2.5 void gpio_set ( unsigned int *pin,* unsigned int *value* )

Set the value of a GPIO pin which is configured as output.

If not already done, this function will set the pin as an output pin.

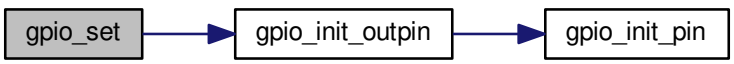
**Parameters**

| | |
|---|---|
| *pin* | - The pin number of the GPIO pin , ranging from 0 to 143) |
| *value* | - The value to set (0 will be low, all other values will map to high) |

**Returns**

none

Here is the call graph for this function:



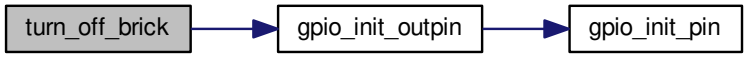
#### 4.32.2.6 void turn_off_brick ( void )

Turn the EV3 off by setting GPIO pin 6 on bank 11 as an output pin.

**Returns**

none

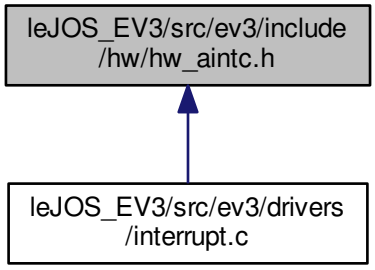
Here is the call graph for this function:




## 4.33 leJOS_EV3/src/ev3/include/hw/hw_aintc.h File Reference

ARM INTC register definitions.

This graph shows which files directly or indirectly include this file:




### Macros

- #define **AINTC_REVID** (0x0)
- #define **AINTC_CR** (0x4)
- #define **AINTC_GER** (0x10)
- #define **AINTC_GNLR** (0x1c)
- #define **AINTC_SISR** (0x20)
- #define **AINTC_SICR** (0x24)
- #define **AINTC_EISR** (0x28)
- #define **AINTC_EICR** (0x2c)
- #define **AINTC_HIEISR** (0x34)
- #define **AINTC_HIEICR** (0x38)
- #define **AINTC_VBR** (0x50)

- #define **AINTC_VSR** (0x54)
- #define **AINTC_VNR** (0x58)
- #define **AINTC_GPIR** (0x80)
- #define **AINTC_GPVR** (0x84)
- #define **AINTC_SRSR**(n) (0x200 + (n ∗ 4))
- #define **AINTC_SECR**(n) (0x280 + (n ∗ 4))
- #define **AINTC_ESR**(n) (0x300 + (n ∗ 4))
- #define **AINTC_ECR**(n) (0x380 + (n ∗ 4))
- #define **AINTC_CMR**(n) (0x400 + (n ∗ 4))
- #define **AINTC_HIPIR**(n) (0x900 + (n ∗ 4))
- #define **AINTC_HINLR**(n) (0x1100 + (n ∗ 4))
- #define **AINTC_HIER** (0x1500)
- #define **AINTC_HIPVR**(n) (0x1600 + (n ∗ 4))
- #define **AINTC_REVID_REV** (0xFFFFFFFFu)
- #define **AINTC_REVID_REV_SHIFT** (0x00000000u)
- #define **AINTC_CR_PRHOLDMODE** (0x00000010u)
- #define **AINTC_CR_PRHOLDMODE_SHIFT** (0x00000004u)
- #define **AINTC_CR_PRHOLDMODE_NO_PRHOLD** (0x00000000u)
- #define **AINTC_CR_PRHOLDMODE_PRHOLD** (0x00000001u)
- #define **AINTC_CR_NESTMODE** (0x0000000Cu)
- #define **AINTC_CR_NESTMODE_SHIFT** (0x00000002u)
- #define **AINTC_CR_NESTMODE_NONEST** (0x00000000u)
- #define **AINTC_CR_NESTMODE_INDIVIDUAL** (0x00000001u)
- #define **AINTC_CR_NESTMODE_GLOBAL** (0x00000002u)
- #define **AINTC_CR_NESTMODE_MANUAL** (0x00000003u)
- #define **AINTC_GER_ENABLE** (0x00000001u)
- #define **AINTC_GER_ENABLE_SHIFT** (0x00000000u)
- #define **AINTC_GNLR_OVERRIDE** (0x80000000u)
- #define **AINTC_GNLR_OVERRIDE_SHIFT** (0x0000001Fu)
- #define **AINTC_GNLR_NESTLVL** (0x000001FFu)
- #define **AINTC_GNLR_NESTLVL_SHIFT** (0x00000000u)
- #define **AINTC_SISR_INDEX** (0x000003FFu)
- #define **AINTC_SISR_INDEX_SHIFT** (0x00000000u)
- #define **AINTC_SICR_INDEX** (0x000003FFu)
- #define **AINTC_SICR_INDEX_SHIFT** (0x00000000u)
- #define **AINTC_EISR_INDEX** (0x000003FFu)
- #define **AINTC_EISR_INDEX_SHIFT** (0x00000000u)
- #define **AINTC_EICR_INDEX** (0x000003FFu)
- #define **AINTC_EICR_INDEX_SHIFT** (0x00000000u)
- #define **AINTC_HIEISR_INDEX** (0x000003FFu)
- #define **AINTC_HIEISR_INDEX_SHIFT** (0x00000000u)
- #define **AINTC_HIDISR_INDEX** (0x000003FFu)
- #define **AINTC_HIDISR_INDEX_SHIFT** (0x00000000u)
- #define **AINTC_VBR_BASE** (0xFFFFFFFFu)
- #define **AINTC_VBR_BASE_SHIFT** (0x00000000u)
- #define **AINTC_VSR_SIZE** (0x000000FFu)
- #define **AINTC_VSR_SIZE_SHIFT** (0x00000000u)
- #define **AINTC_VNR_NULL** (0xFFFFFFFFu)
- #define **AINTC_VNR_NULL_SHIFT** (0x00000000u)
- #define **AINTC_GPIR_NONE** (0x80000000u)
- #define **AINTC_GPIR_NONE_SHIFT** (0x0000001Fu)
- #define **AINTC_GPIR_PRI_INDX** (0x000003FFu)
- #define **AINTC_GPIR_PRI_INDX_SHIFT** (0x00000000u)
- #define **AINTC_GPVR_ADDR** (0xFFFFFFFFu)
- #define **AINTC_GPVR_ADDR_SHIFT** (0x00000000u)

- #define **AINTC_SRSR_RAW_STATUS** (0xFFFFFFFFu)
- #define **AINTC_SRSR_RAW_STATUS_SHIFT** (0x00000000u)
- #define **AINTC_SECR_ENBL_STATUS** (0xFFFFFFFFu)
- #define **AINTC_SECR_ENBL_STATUS_SHIFT** (0x00000000u)
- #define **AINTC_ESR_ENABLE_SHIFT** (0x00000000u)
- #define **AINTC_ECR_DISABLE** (0xFFFFFFFFu)
- #define **AINTC_ECR_DISABLE_SHIFT** (0x00000000u)
- #define **AINTC_CMR_CHNL_NPLUS3** (0xFF000000u)
- #define **AINTC_CMR_CHNL_NPLUS3_SHIFT** (0x00000018u)
- #define **AINTC_CMR_CHNL_NPLUS2** (0x00FF0000u)
- #define **AINTC_CMR_CHNL_NPLUS2_SHIFT** (0x00000010u)
- #define **AINTC_CMR_CHNL_NPLUS1** (0x0000FF00u)
- #define **AINTC_CMR_CHNL_NPLUS1_SHIFT** (0x00000008u)
- #define **AINTC_CMR_CHNL_N** (0x000000FFu)
- #define **AINTC_CMR_CHNL_N_SHIFT** (0x00000000u)
- #define **AINTC_HIPIR_NONE** (0x80000000u)
- #define **AINTC_HIPIR_NONE_SHIFT** (0x0000001Fu)
- #define **AINTC_HIPIR_PRI_INDX** (0x000003FFu)
- #define **AINTC_HIPIR_PRI_INDX_SHIFT** (0x00000000u)
- #define **AINTC_HINLR_OVERRIDE** (0x80000000u)
- #define **AINTC_HINLR_OVERRIDE_SHIFT** (0x0000001Fu)
- #define **AINTC_HINLR_NEST_LVL** (0x000001FFu)
- #define **AINTC_HINLR_NEST_LVL_SHIFT** (0x00000000u)
- #define **AINTC_HINLR_OVERRIDE** (0x80000000u)
- #define **AINTC_HINLR_OVERRIDE_SHIFT** (0x0000001Fu)
- #define **AINTC_HINLR_NEST_LVL** (0x000001FFu)
- #define **AINTC_HINLR_NEST_LVL_SHIFT** (0x00000000u)
- #define **AINTC_HIER_IRQ** (0x00000002u)
- #define **AINTC_HIER_IRQ_SHIFT** (0x00000001u)
- #define **AINTC_HIER_FIQ** (0x00000001u)
- #define **AINTC_HIER_FIQ_SHIFT** (0x00000000u)
- #define **AINTC_HIPVR_ADDR** (0xFFFFFFFFu)
- #define **AINTC_HIPVR_ADDR_SHIFT** (0x00000000u)

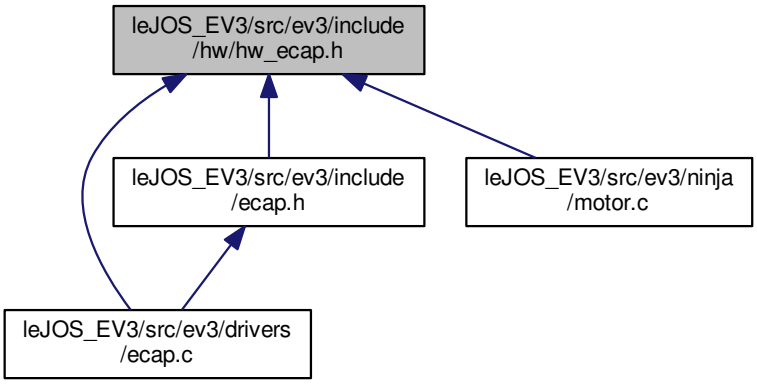
### 4.33.1 Detailed Description

ARM INTC register definitions.

## 4.34 leJOS_EV3/src/ev3/include/hw/hw_ecap.h File Reference

ECAP register definitions.

This graph shows which files directly or indirectly include this file:




## Macros

- #define **ECAP_TSCTR** (0x0)
- #define **ECAP_CTRPHS** (0x4)
- #define **ECAP_CAP1** (0x8)
- #define **ECAP_CAP2** (0xC)
- #define **ECAP_CAP3** (0x10)
- #define **ECAP_CAP4** (0x14)
- #define **ECAP_ECCTL1** (0x28)
- #define **ECAP_ECCTL2** (0x2A)
- #define **ECAP_ECEINT** (0x2C)
- #define **ECAP_ECFLG** (0x2E)
- #define **ECAP_ECCLR** (0x30)
- #define **ECAP_ECFRC** (0x32)
- #define **ECAP_REVID** (0x5C)
- #define **ECAP_TSCTR_TSCTR** (0xFFFFFFFFu)
- #define **ECAP_TSCTR_TSCTR_SHIFT** (0x00000000u)
- #define **ECAP_CTRPHS_CTRPHS** (0xFFFFFFFFu)
- #define **ECAP_CTRPHS_CTRPHS_SHIFT** (0x00000000u)
- #define **ECAP_CAP1_CAP1** (0xFFFFFFFFu)
- #define **ECAP_CAP1_CAP1_SHIFT** (0x00000000u)
- #define **ECAP_CAP2_CAP2** (0xFFFFFFFFu)
- #define **ECAP_CAP2_CAP2_SHIFT** (0x00000000u)
- #define **ECAP_CAP3_CAP3** (0xFFFFFFFFu)
- #define **ECAP_CAP3_CAP3_SHIFT** (0x00000000u)
- #define **ECAP_CAP4_CAP4** (0xFFFFFFFFu)
- #define **ECAP_CAP4_CAP4_SHIFT** (0x00000000u)
- #define **ECAP_ECCTL1_FREE_SOFT** (0xC000u)
- #define **ECAP_ECCTL1_FREE_SOFT_SHIFT** (0x000Eu)
- #define **ECAP_ECCTL1_FREE_SOFT_STOP** (0x0000u)
- #define **ECAP_ECCTL1_FREE_SOFT_RUNUNTIL0** (0x0001u)
- #define **ECAP_ECCTL1_PRESCALE** (0x3E00u)
- #define **ECAP_ECCTL1_PRESCALE_SHIFT** (0x0009u)
- #define **ECAP_ECCTL1_CAPLDEN** (0x0100u)

- #define **ECAP_ECCTL1_CAPLDEN_SHIFT** (0x0008u)
- #define **ECAP_ECCTL1_CTRRST4** (0x0080u)
- #define **ECAP_ECCTL1_CTRRST4_SHIFT** (0x0007u)
- #define **ECAP_ECCTL1_CTRRST4_NORESET** (0x0000u)
- #define **ECAP_ECCTL1_CTRRST4_RESET** (0x0001u)
- #define **ECAP_ECCTL1_CAP4POL** (0x0040u)
- #define **ECAP_ECCTL1_CAP4POL_SHIFT** (0x0006u)
- #define **ECAP_ECCTL1_CTRRST3** (0x0020u)
- #define **ECAP_ECCTL1_CTRRST3_SHIFT** (0x0005u)
- #define **ECAP_ECCTL1_CAP3POL** (0x0010u)
- #define **ECAP_ECCTL1_CAP3POL_SHIFT** (0x0004u)
- #define **ECAP_ECCTL1_CTRRST2** (0x0008u)
- #define **ECAP_ECCTL1_CTRRST2_SHIFT** (0x0003u)
- #define **ECAP_ECCTL1_CAP2POL** (0x0004u)
- #define **ECAP_ECCTL1_CAP2POL_SHIFT** (0x0002u)
- #define **ECAP_ECCTL1_CTRRST1** (0x0002u)
- #define **ECAP_ECCTL1_CTRRST1_SHIFT** (0x0001u)
- #define **ECAP_ECCTL1_CAP1POL** (0x0001u)
- #define **ECAP_ECCTL1_CAP1POL_SHIFT** (0x0000u)
- #define **ECAP_ECCTL2_APWMPOL** (0x0400u)
- #define **ECAP_ECCTL2_APWMPOL_SHIFT** (0x000Au)
- #define **ECAP_ECCTL2_CAP_APWM** (0x0200u)
- #define **ECAP_ECCTL2_CAP_APWM_SHIFT** (0x0009u)
- #define **ECAP_ECCTL2_SWSYNC** (0x0100u)
- #define **ECAP_ECCTL2_SWSYNC_SHIFT** (0x0008u)
- #define **ECAP_ECCTL2_SYNCO_SEL** (0x00C0u)
- #define **ECAP_ECCTL2_SYNCO_SEL_SHIFT** (0x0006u)
- #define **ECAP_ECCTL2_SYNCI_EN** (0x0020u)
- #define **ECAP_ECCTL2_SYNCI_EN_SHIFT** (0x0005u)
- #define **ECAP_ECCTL2_TSCTRSTOP** (0x0010u)
- #define **ECAP_ECCTL2_TSCTRSTOP_SHIFT** (0x0004u)
- #define **ECAP_ECCTL2_RE_ARM** (0x0008u)
- #define **ECAP_ECCTL2_RE_ARM_SHIFT** (0x0003u)
- #define **ECAP_ECCTL2_STOP_WRAP** (0x0006u)
- #define **ECAP_ECCTL2_STOP_WRAP_SHIFT** (0x0001u)
- #define **ECAP_ECCTL2_STOP_WRAP_CAP1** (0x0000u)
- #define **ECAP_ECCTL2_STOP_WRAP_CAP2** (0x0001u)
- #define **ECAP_ECCTL2_STOP_WRAP_CAP3** (0x0002u)
- #define **ECAP_ECCTL2_STOP_WRAP_CAP4** (0x0003u)
- #define **ECAP_ECCTL2_CONT_ONESHT** (0x0001u)
- #define **ECAP_ECCTL2_CONT_ONESHT_SHIFT** (0x0000u)
- #define **ECAP_ECEINT_CTR_CMP** (0x0080u)
- #define **ECAP_ECEINT_CTR_CMP_SHIFT** (0x0007u)
- #define **ECAP_ECEINT_CTR_PRD** (0x0040u)
- #define **ECAP_ECEINT_CTR_PRD_SHIFT** (0x0006u)
- #define **ECAP_ECEINT_CTROVF** (0x0020u)
- #define **ECAP_ECEINT_CTROVF_SHIFT** (0x0005u)
- #define **ECAP_ECEINT_CEVT4** (0x0010u)
- #define **ECAP_ECEINT_CEVT4_SHIFT** (0x0004u)
- #define **ECAP_ECEINT_CEVT3** (0x0008u)
- #define **ECAP_ECEINT_CEVT3_SHIFT** (0x0003u)
- #define **ECAP_ECEINT_CEVT2** (0x0004u)
- #define **ECAP_ECEINT_CEVT2_SHIFT** (0x0002u)
- #define **ECAP_ECEINT_CEVT1** (0x0002u)
- #define **ECAP_ECEINT_CEVT1_SHIFT** (0x0001u)

- #define **ECAP_ECFLG_CTR_CMP** (0x0080u)
- #define **ECAP_ECFLG_CTR_CMP_SHIFT** (0x0007u)
- #define **ECAP_ECFLG_CTR_PRD** (0x0040u)
- #define **ECAP_ECFLG_CTR_PRD_SHIFT** (0x0006u)
- #define **ECAP_ECFLG_CTROVF** (0x0020u)
- #define **ECAP_ECFLG_CTROVF_SHIFT** (0x0005u)
- #define **ECAP_ECFLG_CEVT4** (0x0010u)
- #define **ECAP_ECFLG_CEVT4_SHIFT** (0x0004u)
- #define **ECAP_ECFLG_CEVT3** (0x0008u)
- #define **ECAP_ECFLG_CEVT3_SHIFT** (0x0003u)
- #define **ECAP_ECFLG_CEVT2** (0x0004u)
- #define **ECAP_ECFLG_CEVT2_SHIFT** (0x0002u)
- #define **ECAP_ECFLG_CEVT1** (0x0002u)
- #define **ECAP_ECFLG_CEVT1_SHIFT** (0x0001u)
- #define **ECAP_ECFLG_INT** (0x0001u)
- #define **ECAP_ECFLG_INT_SHIFT** (0x0000u)
- #define **ECAP_ECCLR_CTR_CMP** (0x0080u)
- #define **ECAP_ECCLR_CTR_CMP_SHIFT** (0x0007u)
- #define **ECAP_ECCLR_CTR_PRD** (0x0040u)
- #define **ECAP_ECCLR_CTR_PRD_SHIFT** (0x0006u)
- #define **ECAP_ECCLR_CTROVF** (0x0020u)
- #define **ECAP_ECCLR_CTROVF_SHIFT** (0x0005u)
- #define **ECAP_ECCLR_CEVT4** (0x0010u)
- #define **ECAP_ECCLR_CEVT4_SHIFT** (0x0004u)
- #define **ECAP_ECCLR_CEVT3** (0x0008u)
- #define **ECAP_ECCLR_CEVT3_SHIFT** (0x0003u)
- #define **ECAP_ECCLR_CEVT2** (0x0004u)
- #define **ECAP_ECCLR_CEVT2_SHIFT** (0x0002u)
- #define **ECAP_ECCLR_CEVT1** (0x0002u)
- #define **ECAP_ECCLR_CEVT1_SHIFT** (0x0001u)
- #define **ECAP_ECCLR_INT** (0x0001u)
- #define **ECAP_ECCLR_INT_SHIFT** (0x0000u)
- #define **ECAP_ECFRC_CTR_CMP** (0x0080u)
- #define **ECAP_ECFRC_CTR_CMP_SHIFT** (0x0007u)
- #define **ECAP_ECFRC_CTR_PRD** (0x0040u)
- #define **ECAP_ECFRC_CTR_PRD_SHIFT** (0x0006u)
- #define **ECAP_ECFRC_CTROVF** (0x0020u)
- #define **ECAP_ECFRC_CTROVF_SHIFT** (0x0005u)
- #define **ECAP_ECFRC_CEVT4** (0x0010u)
- #define **ECAP_ECFRC_CEVT4_SHIFT** (0x0004u)
- #define **ECAP_ECFRC_CEVT3** (0x0008u)
- #define **ECAP_ECFRC_CEVT3_SHIFT** (0x0003u)
- #define **ECAP_ECFRC_CEVT2** (0x0004u)
- #define **ECAP_ECFRC_CEVT2_SHIFT** (0x0002u)
- #define **ECAP_ECFRC_CEVT1** (0x0002u)
- #define **ECAP_ECFRC_CEVT1_SHIFT** (0x0001u)
- #define **ECAP_REVID_REV** (0xFFFFFFFFu)
- #define **ECAP_REVID_REV_SHIFT** (0x00000000u)

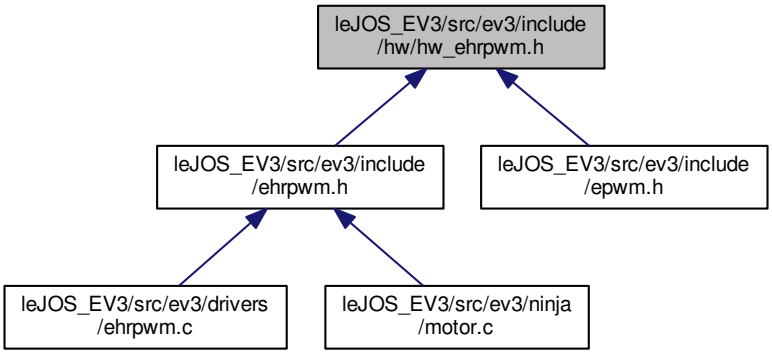
### 4.34.1 Detailed Description

ECAP register definitions.

## 4.35 leJOS_EV3/src/ev3/include/hw/hw_ehrpwm.h File Reference

EHRPWM register definitions.

This graph shows which files directly or indirectly include this file:




### Macros

- #define **EHRPWM_TBCTL** (0x0)
- #define **EHRPWM_TBSTS** (0x2)
- #define **EHRPWM_TBPHSHR** (0x4)
- #define **EHRPWM_TBPHS** (0x6)
- #define **EHRPWM_TBCTR** (0x8)
- #define **EHRPWM_TBPRD** (0xA)
- #define **EHRPWM_CMPCTL** (0xE)
- #define **EHRPWM_CMPAHR** (0x10)
- #define **EHRPWM_CMPA** (0x12)
- #define **EHRPWM_CMPB** (0x14)
- #define **EHRPWM_AQCTLA** (0x16)
- #define **EHRPWM_AQCTLB** (0x18)
- #define **EHRPWM_AQSFRC** (0x1A)
- #define **EHRPWM_AQCSFRC** (0x1C)
- #define **EHRPWM_DBCTL** (0x1E)
- #define **EHRPWM_DBRED** (0x20)
- #define **EHRPWM_DBFED** (0x22)
- #define **EHRPWM_TZSEL** (0x24)
- #define **EHRPWM_TZCTL** (0x28)
- #define **EHRPWM_TZEINT** (0x2A)
- #define **EHRPWM_TZFLG** (0x2C)
- #define **EHRPWM_TZCLR** (0x2E)
- #define **EHRPWM_TZFRC** (0x30)
- #define **EHRPWM_ETSEL** (0x32)
- #define **EHRPWM_ETPS** (0x34)
- #define **EHRPWM_ETFLG** (0x36)
- #define **EHRPWM_ETCLR** (0x38)
- #define **EHRPWM_ETFRC** (0x3A)
- #define **EHRPWM_PCCTL** (0x3C)

- #define **EHRPWM_TBCTL_FREE_SOFT** (0xC000u)
- #define **EHRPWM_TBCTL_FREE_SOFT_SHIFT** (0x000Eu)
- #define **EHRPWM_TBCTL_PHSDIR** (0x2000u)
- #define **EHRPWM_TBCTL_PHSDIR_SHIFT** (0x000Du)
- #define **EHRPWM_TBCTL_CLKDIV** (0x1C00u)
- #define **EHRPWM_TBCTL_CLKDIV_SHIFT** (0x000Au)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY1** (0x0000u)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY2** (0x0001u)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY4** (0x0002u)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY8** (0x0003u)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY16** (0x0004u)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY32** (0x0005u)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY64** (0x0006u)
- #define **EHRPWM_TBCTL_CLKDIV_DIVBY128** (0x0007u)
- #define **EHRPWM_TBCTL_HSPCLKDIV** (0x0380u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_SHIFT** (0x0007u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY1** (0x0000u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY2** (0x0001u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY4** (0x0002u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY6** (0x0003u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY8** (0x0004u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY10** (0x0005u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY12** (0x0006u)
- #define **EHRPWM_TBCTL_HSPCLKDIV_DIVBY14** (0x0007u)
- #define **EHRPWM_TBCTL_SWFSYNC** (0x0040u)
- #define **EHRPWM_TBCTL_SWFSYNC_SHIFT** (0x0006u)
- #define **EHRPWM_TBCTL_SYNCOSEL** (0x0030u)
- #define **EHRPWM_TBCTL_SYNCOSEL_SHIFT** (0x0004u)
- #define **EHRPWM_TBCTL_SYNCOSEL_EPWMXSYNCI** (0x0000u)
- #define **EHRPWM_TBCTL_SYNCOSEL_TBCTRZERO** (0x0001u)
- #define **EHRPWM_TBCTL_SYNCOSEL_TBCTRCMPB** (0x0002u)
- #define **EHRPWM_TBCTL_SYNCOSEL_DISABLE** (0x0003u)
- #define **EHRPWM_TBCTL_PRDLD** (0x0008u)
- #define **EHRPWM_TBCTL_PRDLD_SHIFT** (0x0003u)
- #define **EHRPWM_TBCTL_PHSEN** (0x0004u)
- #define **EHRPWM_TBCTL_PHSEN_SHIFT** (0x0002u)
- #define **EHRPWM_TBCTL_CTRMODE** (0x0003u)
- #define **EHRPWM_TBCTL_CTRMODE_SHIFT** (0x0000u)
- #define **EHRPWM_TBCTL_CTRMODE_UP** (0x0000u)
- #define **EHRPWM_TBCTL_CTRMODE_DOWN** (0x0001u)
- #define **EHRPWM_TBCTL_CTRMODE_UPDOWN** (0x0002u)
- #define **EHRPWM_TBCTL_CTRMODE_STOPFREEZE** (0x0003u)
- #define **EHRPWM_TBSTS_CTRMAX** (0x0004u)
- #define **EHRPWM_TBSTS_CTRMAX_SHIFT** (0x0002u)
- #define **EHRPWM_TBSTS_SYNCI** (0x0002u)
- #define **EHRPWM_TBSTS_SYNCI_SHIFT** (0x0001u)
- #define **EHRPWM_TBSTS_CTRDIR** (0x0001u)
- #define **EHRPWM_TBSTS_CTRDIR_SHIFT** (0x0000u)
- #define **EHRPWM_TBPHSHR_TBPHSHR** (0xFF00u)
- #define **EHRPWM_TBPHSHR_TBPHSHR_SHIFT** (0x0008u)
- #define **EHRPWM_TBPHS_TBPHS** (0xFFFFu)
- #define **EHRPWM_TBPHS_TBPHS_SHIFT** (0x0000u)
- #define **EHRPWM_TBCTR_TBCTR** (0xFFFFu)
- #define **EHRPWM_TBCTR_TBCTR_SHIFT** (0x0000u)
- #define **EHRPWM_TBPRD_TBPRD** (0xFFFFu)

- #define **EHRPWM_TBPRD_TBPRD_SHIFT** (0x0000u)
- #define **EHRPWM_CMPCTL_SHDWBFULL** (0x0200u)
- #define **EHRPWM_CMPCTL_SHDWBFULL_SHIFT** (0x0009u)
- #define **EHRPWM_CMPCTL_SHDWAFULL** (0x0100u)
- #define **EHRPWM_CMPCTL_SHDWAFULL_SHIFT** (0x0008u)
- #define **EHRPWM_CMPCTL_SHDWBMODE** (0x0040u)
- #define **EHRPWM_CMPCTL_SHDWBMODE_SHIFT** (0x0006u)
- #define **EHRPWM_CMPCTL_SHDWAMODE** (0x0010u)
- #define **EHRPWM_CMPCTL_SHDWAMODE_SHIFT** (0x0004u)
- #define **EHRPWM_CMPCTL_LOADBMODE** (0x000Cu)
- #define **EHRPWM_CMPCTL_LOADBMODE_SHIFT** (0x0002u)
- #define **EHRPWM_CMPCTL_LOADBMODE_TBCTRZERO** (0x0000u)
- #define **EHRPWM_CMPCTL_LOADBMODE_TBCTRPRD** (0x0001u)
- #define **EHRPWM_CMPCTL_LOADBMODE_ZEROORPRD** (0x0002u)
- #define **EHRPWM_CMPCTL_LOADBMODE_FREEZE** (0x0003u)
- #define **EHRPWM_CMPCTL_LOADAMODE** (0x0003u)
- #define **EHRPWM_CMPCTL_LOADAMODE_SHIFT** (0x0000u)
- #define **EHRPWM_CMPCTL_LOADAMODE_TBCTRZERO** (0x0000u)
- #define **EHRPWM_CMPCTL_LOADAMODE_TBCTRPRD** (0x0001u)
- #define **EHRPWM_CMPCTL_LOADAMODE_ZEROORPRD** (0x0002u)
- #define **EHRPWM_CMPCTL_LOADAMODE_FREEZE** (0x0003u)
- #define **EHRPWM_CMPAHR_CMPAHR** (0xFF00u)
- #define **EHRPWM_CMPAHR_CMPAHR_SHIFT** (0x0008u)
- #define **EHRPWM_CMPA_CMPA** (0xFFFFu)
- #define **EHRPWM_CMPA_CMPA_SHIFT** (0x0000u)
- #define **EHRPWM_CMPB_CMPB** (0xFFFFu)
- #define **EHRPWM_CMPB_CMPB_SHIFT** (0x0000u)
- #define **EHRPWM_AQCTLA_CBD** (0x0C00u)
- #define **EHRPWM_AQCTLA_CBD_SHIFT** (0x000Au)
- #define **EHRPWM_AQCTLA_CBD_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLA_CBD_EPWMXALOW** (0x0001u)
- #define **EHRPWM_AQCTLA_CBD_EPWMXAHIGH** (0x0002u)
- #define **EHRPWM_AQCTLA_CBD_EPWMXATOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLA_CBU** (0x0300u)
- #define **EHRPWM_AQCTLA_CBU_SHIFT** (0x0008u)
- #define **EHRPWM_AQCTLA_CBU_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLA_CBU_EPWMXALOW** (0x0001u)
- #define **EHRPWM_AQCTLA_CBU_EPWMXAHIGH** (0x0002u)
- #define **EHRPWM_AQCTLA_CBU_EPWMXATOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLA_CAD** (0x00C0u)
- #define **EHRPWM_AQCTLA_CAD_SHIFT** (0x0006u)
- #define **EHRPWM_AQCTLA_CAD_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLA_CAD_EPWMXALOW** (0x0001u)
- #define **EHRPWM_AQCTLA_CAD_EPWMXAHIGH** (0x0002u)
- #define **EHRPWM_AQCTLA_CAD_EPWMXATOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLA_CAU** (0x0030u)
- #define **EHRPWM_AQCTLA_CAU_SHIFT** (0x0004u)
- #define **EHRPWM_AQCTLA_CAU_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLA_CAU_EPWMXALOW** (0x0001u)
- #define **EHRPWM_AQCTLA_CAU_EPWMXAHIGH** (0x0002u)
- #define **EHRPWM_AQCTLA_CAU_EPWMXATOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLA_PRD** (0x000Cu)
- #define **EHRPWM_AQCTLA_PRD_SHIFT** (0x0002u)
- #define **EHRPWM_AQCTLA_PRD_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLA_PRD_EPWMXALOW** (0x0001u)

- #define **EHRPWM_AQCTLA_PRD_EPWMXAHIGH** (0x0002u)
- #define **EHRPWM_AQCTLA_PRD_EPWMXATOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLA_ZRO** (0x0003u)
- #define **EHRPWM_AQCTLA_ZRO_SHIFT** (0x0000u)
- #define **EHRPWM_AQCTLA_ZRO_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLA_ZRO_EPWMXALOW** (0x0001u)
- #define **EHRPWM_AQCTLA_ZRO_EPWMXAHIGH** (0x0002u)
- #define **EHRPWM_AQCTLA_ZRO_EPWMXATOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLB_CBD** (0x0C00u)
- #define **EHRPWM_AQCTLB_CBD_SHIFT** (0x000Au)
- #define **EHRPWM_AQCTLB_CBD_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLB_CBD_EPWMXBLOW** (0x0001u)
- #define **EHRPWM_AQCTLB_CBD_EPWMXBHIGH** (0x0002u)
- #define **EHRPWM_AQCTLB_CBD_EPWMXBTOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLB_CBU** (0x0300u)
- #define **EHRPWM_AQCTLB_CBU_SHIFT** (0x0008u)
- #define **EHRPWM_AQCTLB_CBU_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLB_CBU_EPWMXBLOW** (0x0001u)
- #define **EHRPWM_AQCTLB_CBU_EPWMXBHIGH** (0x0002u)
- #define **EHRPWM_AQCTLB_CBU_EPWMXBTOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLB_CAD** (0x00C0u)
- #define **EHRPWM_AQCTLB_CAD_SHIFT** (0x0006u)
- #define **EHRPWM_AQCTLB_CAD_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLB_CAD_EPWMXBLOW** (0x0001u)
- #define **EHRPWM_AQCTLB_CAD_EPWMXBHIGH** (0x0002u)
- #define **EHRPWM_AQCTLB_CAD_EPWMXBTOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLB_CAU** (0x0030u)
- #define **EHRPWM_AQCTLB_CAU_SHIFT** (0x0004u)
- #define **EHRPWM_AQCTLB_CAU_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLB_CAU_EPWMXBLOW** (0x0001u)
- #define **EHRPWM_AQCTLB_CAU_EPWMXBHIGH** (0x0002u)
- #define **EHRPWM_AQCTLB_CAU_EPWMXBTOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLB_PRD** (0x000Cu)
- #define **EHRPWM_AQCTLB_PRD_SHIFT** (0x0002u)
- #define **EHRPWM_AQCTLB_PRD_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLB_PRD_EPWMXBLOW** (0x0001u)
- #define **EHRPWM_AQCTLB_PRD_EPWMXBHIGH** (0x0002u)
- #define **EHRPWM_AQCTLB_PRD_EPWMXBTOGGLE** (0x0003u)
- #define **EHRPWM_AQCTLB_ZRO** (0x0003u)
- #define **EHRPWM_AQCTLB_ZRO_SHIFT** (0x0000u)
- #define **EHRPWM_AQCTLB_ZRO_DONOTHING** (0x0000u)
- #define **EHRPWM_AQCTLB_ZRO_EPWMXBLOW** (0x0001u)
- #define **EHRPWM_AQCTLB_ZRO_EPWMXBHIGH** (0x0002u)
- #define **EHRPWM_AQCTLB_ZRO_EPWMXBTOGGLE** (0x0003u)
- #define **EHRPWM_AQSFRC_RLDCSF** (0x00C0u)
- #define **EHRPWM_AQSFRC_RLDCSF_SHIFT** (0x0006u)
- #define **EHRPWM_AQSFRC_RLDCSF_EVTCTRZERO** (0x0000u)
- #define **EHRPWM_AQSFRC_RLDCSF_EVTCTRPRD** (0x0001u)
- #define **EHRPWM_AQSFRC_RLDCSF_ZEROORPRD** (0x0002u)
- #define **EHRPWM_AQSFRC_RLDCSF_IMMEDIATE** (0x0003u)
- #define **EHRPWM_AQSFRC_OTSFB** (0x0020u)
- #define **EHRPWM_AQSFRC_OTSFB_SHIFT** (0x0005u)
- #define **EHRPWM_AQSFRC_OTSFB_NOEFFECT** (0x0000u)
- #define **EHRPWM_AQSFRC_OTSFB_EVENT** (0x0001u)
- #define **EHRPWM_AQSFRC_ACTSFB** (0x0018u)

- #define **EHRPWM_AQSFRC_ACTSFB_SHIFT** (0x0003u)
- #define **EHRPWM_AQSFRC_ACTSFB_DONOTHING** (0x0000u)
- #define **EHRPWM_AQSFRC_ACTSFB_CLEAR** (0x0001u)
- #define **EHRPWM_AQSFRC_ACTSFB_SET** (0x0002u)
- #define **EHRPWM_AQSFRC_ACTSFB_TOGGLE** (0x0003u)
- #define **EHRPWM_AQSFRC_OTSFA** (0x0004u)
- #define **EHRPWM_AQSFRC_OTSFA_SHIFT** (0x0002u)
- #define **EHRPWM_AQSFRC_OTSFA_NOEFFECT** (0x0000u)
- #define **EHRPWM_AQSFRC_OTSFA_EVENT** (0x0001u)
- #define **EHRPWM_AQSFRC_ACTSFA** (0x0003u)
- #define **EHRPWM_AQSFRC_ACTSFA_SHIFT** (0x0000u)
- #define **EHRPWM_AQSFRC_ACTSFA_DONOTHING** (0x0000u)
- #define **EHRPWM_AQSFRC_ACTSFA_CLEAR** (0x0001u)
- #define **EHRPWM_AQSFRC_ACTSFA_SET** (0x0002u)
- #define **EHRPWM_AQSFRC_ACTSFA_TOGGLE** (0x0003u)
- #define **EHRPWM_AQCSFRC_CSFB** (0x000Cu)
- #define **EHRPWM_AQCSFRC_CSFB_SHIFT** (0x0002u)
- #define **EHRPWM_AQCSFRC_CSFB_LOW** (0x0001u)
- #define **EHRPWM_AQCSFRC_CSFB_HIGH** (0x0002u)
- #define **EHRPWM_AQCSFRC_CSFA** (0x0003u)
- #define **EHRPWM_AQCSFRC_CSFA_SHIFT** (0x0000u)
- #define **EHRPWM_AQCSFRC_CSFA_LOW** (0x0001u)
- #define **EHRPWM_AQCSFRC_CSFA_HIGH** (0x0002u)
- #define **EHRPWM_DBCTL_IN_MODE** (0x0030u)
- #define **EHRPWM_DBCTL_IN_MODE_SHIFT** (0x0004u)
- #define **EHRPWM_DBCTL_IN_MODE_AREDAFED** (0x0000u)
- #define **EHRPWM_DBCTL_IN_MODE_BREDAFED** (0x0001u)
- #define **EHRPWM_DBCTL_IN_MODE_AREDBFED** (0x0002u)
- #define **EHRPWM_DBCTL_IN_MODE_BREDBFED** (0x0003u)
- #define **EHRPWM_DBCTL_POLSEL** (0x000Cu)
- #define **EHRPWM_DBCTL_POLSEL_SHIFT** (0x0002u)
- #define **EHRPWM_DBCTL_POLSEL_ACTIVEHIGH** (0x0000u)
- #define **EHRPWM_DBCTL_POLSEL_ALC** (0x0001u)
- #define **EHRPWM_DBCTL_POLSEL_AHC** (0x0002u)
- #define **EHRPWM_DBCTL_POLSEL_ACTIVELOW** (0x0003u)
- #define **EHRPWM_DBCTL_OUT_MODE** (0x0003u)
- #define **EHRPWM_DBCTL_OUT_MODE_SHIFT** (0x0000u)
- #define **EHRPWM_DBCTL_OUT_MODE_BYPASS** (0x0000u)
- #define **EHRPWM_DBCTL_OUT_MODE_NOREDBFED** (0x0001u)
- #define **EHRPWM_DBCTL_OUT_MODE_AREDNOFED** (0x0002u)
- #define **EHRPWM_DBCTL_OUT_MODE_AREDBFED** (0x0003u)
- #define **EHRPWM_DBRED_DEL** (0x03FFu)
- #define **EHRPWM_DBRED_DEL_SHIFT** (0x0000u)
- #define **EHRPWM_DBFED_DEL** (0x03FFu)
- #define **EHRPWM_DBFED_DEL_SHIFT** (0x0000u)
- #define **EHRPWM_TZSEL_OSHT1** (0x0100u)
- #define **EHRPWM_TZSEL_OSHT1_SHIFT** (0x0008u)
- #define **EHRPWM_TZSEL_CBC1** (0x0001u)
- #define **EHRPWM_TZSEL_CBC1_SHIFT** (0x0000u)
- #define **EHRPWM_TZCTL_TZB** (0x000Cu)
- #define **EHRPWM_TZCTL_TZB_SHIFT** (0x0002u)
- #define **EHRPWM_TZCTL_TZB_TRISTATE** (0x0000u)
- #define **EHRPWM_TZCTL_TZB_FORCEHIGH** (0x0001u)
- #define **EHRPWM_TZCTL_TZB_FORCELOW** (0x0002u)
- #define **EHRPWM_TZCTL_TZB_DONOTHING** (0x0003u)

- #define **EHRPWM_TZCTL_TZA** (0x0003u)
- #define **EHRPWM_TZCTL_TZA_SHIFT** (0x0000u)
- #define **EHRPWM_TZCTL_TZA_TRISTATE** (0x0000u)
- #define **EHRPWM_TZCTL_TZA_FORCEHIGH** (0x0001u)
- #define **EHRPWM_TZCTL_TZA_FORCELOW** (0x0002u)
- #define **EHRPWM_TZCTL_TZA_DONOTHING** (0x0003u)
- #define **EHRPWM_TZEINT_OST** (0x0004u)
- #define **EHRPWM_TZEINT_OST_SHIFT** (0x0002u)
- #define **EHRPWM_TZEINT_CBC** (0x0002u)
- #define **EHRPWM_TZEINT_CBC_SHIFT** (0x0001u)
- #define **EHRPWM_TZFLG_OST** (0x0004u)
- #define **EHRPWM_TZFLG_OST_SHIFT** (0x0002u)
- #define **EHRPWM_TZFLG_CBC** (0x0002u)
- #define **EHRPWM_TZFLG_CBC_SHIFT** (0x0001u)
- #define **EHRPWM_TZFLG_INT** (0x0001u)
- #define **EHRPWM_TZFLG_INT_SHIFT** (0x0000u)
- #define **EHRPWM_TZCLR_OST** (0x0004u)
- #define **EHRPWM_TZCLR_OST_SHIFT** (0x0002u)
- #define **EHRPWM_TZCLR_CBC** (0x0002u)
- #define **EHRPWM_TZCLR_CBC_SHIFT** (0x0001u)
- #define **EHRPWM_TZCLR_INT** (0x0001u)
- #define **EHRPWM_TZCLR_INT_SHIFT** (0x0000u)
- #define **EHRPWM_TZFRC_OST** (0x0004u)
- #define **EHRPWM_TZFRC_OST_SHIFT** (0x0002u)
- #define **EHRPWM_TZFRC_CBC** (0x0002u)
- #define **EHRPWM_TZFRC_CBC_SHIFT** (0x0001u)
- #define **EHRPWM_ETSEL_INTEN** (0x0008u)
- #define **EHRPWM_ETSEL_INTEN_SHIFT** (0x0003u)
- #define **EHRPWM_ETSEL_INTSEL** (0x0007u)
- #define **EHRPWM_ETSEL_INTSEL_SHIFT** (0x0000u)
- #define **EHRPWM_ETSEL_INTSEL_TBCTREQUZERO** (0x0001u)
- #define **EHRPWM_ETSEL_INTSEL_TBCTREQUPRD** (0x0002u)
- #define **EHRPWM_ETSEL_INTSEL_TBCTREQUCMPAINC** (0x0004u)
- #define **EHRPWM_ETSEL_INTSEL_TBCTREQUCMPADEC** (0x0005u)
- #define **EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBINC** (0x0006u)
- #define **EHRPWM_ETSEL_INTSEL_TBCTREQUCMPBDEC** (0x0007u)
- #define **EHRPWM_ETPS_INTCNT** (0x000Cu)
- #define **EHRPWM_ETPS_INTCNT_SHIFT** (0x0002u)
- #define **EHRPWM_ETPS_INTCNT_NOEVENTS** (0x0000u)
- #define **EHRPWM_ETPS_INTCNT_ONEEVENT** (0x0001u)
- #define **EHRPWM_ETPS_INTCNT_TWOEVENTS** (0x0002u)
- #define **EHRPWM_ETPS_INTCNT_THREEEVENTS** (0x0003u)
- #define **EHRPWM_ETPS_INTPRD** (0x0003u)
- #define **EHRPWM_ETPS_INTPRD_SHIFT** (0x0000u)
- #define **EHRPWM_ETPS_INTPRD_FIRSTEVENT** (0x0001u)
- #define **EHRPWM_ETPS_INTPRD_SECONDEVENT** (0x0002u)
- #define **EHRPWM_ETPS_INTPRD_THIRDEVENT** (0x0003u)
- #define **EHRPWM_ETFLG_INT** (0x0001u)
- #define **EHRPWM_ETFLG_INT_SHIFT** (0x0000u)
- #define **EHRPWM_ETCLR_INT** (0x0001u)
- #define **EHRPWM_ETCLR_INT_SHIFT** (0x0000u)
- #define **EHRPWM_ETCLR_INT_NOEFFECT** (0x0000u)
- #define **EHRPWM_ETCLR_INT_CLEAR** (0x0001u)
- #define **EHRPWM_ETFRC_INT** (0x0001u)
- #define **EHRPWM_ETFRC_INT_SHIFT** (0x0000u)

- #define **EHRPWM_PCCTL_CHPDUTY** (0x0700u)
- #define **EHRPWM_PCCTL_CHPDUTY_SHIFT** (0x0008u)
- #define **EHRPWM_PCCTL_CHPDUTY_1DIV8** (0x0000u)
- #define **EHRPWM_PCCTL_CHPDUTY_2DIV8** (0x0001u)
- #define **EHRPWM_PCCTL_CHPDUTY_3DIV8** (0x0002u)
- #define **EHRPWM_PCCTL_CHPDUTY_4DIV8** (0x0003u)
- #define **EHRPWM_PCCTL_CHPDUTY_5DIV8** (0x0004u)
- #define **EHRPWM_PCCTL_CHPDUTY_6DIV8** (0x0005u)
- #define **EHRPWM_PCCTL_CHPDUTY_7DIV8** (0x0006u)
- #define **EHRPWM_PCCTL_CHPDUTY_RESERVED** (0x0007u)
- #define **EHRPWM_PCCTL_CHPFREQ** (0x00E0u)
- #define **EHRPWM_PCCTL_CHPFREQ_SHIFT** (0x0005u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY1** (0x0000u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY2** (0x0001u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY3** (0x0002u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY4** (0x0003u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY5** (0x0004u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY6** (0x0005u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY7** (0x0006u)
- #define **EHRPWM_PCCTL_CHPFREQ_DIVBY8** (0x0007u)
- #define **EHRPWM_PCCTL_OSHTWTH** (0x001Eu)
- #define **EHRPWM_PCCTL_OSHTWTH_SHIFT** (0x0001u)
- #define **EHRPWM_PCCTL_CHPEN** (0x0001u)
- #define **EHRPWM_PCCTL_CHPEN_SHIFT** (0x0000u)
- #define **EHRPWM_HR_HRLOAD** (0x0008u)
- #define **EHRPWM_HR_HRLOAD_SHIFT** (0x0003u)
- #define **EHRPWM_HR_HRLOAD_CTR_ZERO** (0x0000u)
- #define **EHRPWM_HR_HRLOAD_CTR_PRD** (0x0008u)
- #define **EHRPWM_HR_CTLMODE** (0x0004u)
- #define **EHRPWM_HR_CTLMODE_SHIFT** (0x0002u)
- #define **EHRPWM_HR_CTLMODE_CMPAHR** (0x0000u)
- #define **EHRPWM_HR_CTLMODE_TBPHSHR** (0x0004u)
- #define **EHRPWM_HR_EDGEMODE** (0x0003u)
- #define **EHRPWM_HR_EDGEMODE_SHIFT** (0x0000u)
- #define **EHRPWM_HR_EDGEMODE_DISABLE** (0x0000u)
- #define **EHRPWM_HR_EDGEMODE_RAISING** (0x0001u)
- #define **EHRPWM_HR_EDGEMODE_FALLING** (0x0002u)
- #define **EHRPWM_HR_EDGEMODE_BOTH** (0x0003u)
- #define **EHRPWM_REVID_REV** (0xFFFFFFFFu)
- #define **EHRPWM_REVID_REV_SHIFT** (0x00000000u)

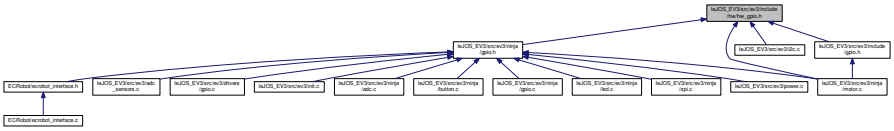
### 4.35.1 Detailed Description

EHRPWM register definitions.

## 4.36 leJOS_EV3/src/ev3/include/hw/hw_gpio.h File Reference

GPIO register definitions.

This graph shows which files directly or indirectly include this file:



## Macros

- #define **GPIO_REVID** (0x0)
- #define **GPIO_BINTEN** (0x8)
- #define **GPIO_DIR**(n) (0x10 + (0x28 * n))
- #define **GPIO_OUT_DATA**(n) (0x14 + (0x28 * n))
- #define **GPIO_SET_DATA**(n) (0x18 + (0x28 * n))
- #define **GPIO_CLR_DATA**(n) (0x1C + (0x28 * n))
- #define **GPIO_IN_DATA**(n) (0x20 + (0x28 * n))
- #define **GPIO_SET_RIS_TRIG**(n) (0x24 + (0x28 * n))
- #define **GPIO_CLR_RIS_TRIG**(n) (0x28 + (0x28 * n))
- #define **GPIO_SET_FAL_TRIG**(n) (0x2C + (0x28 * n))
- #define **GPIO_CLR_FAL_TRIG**(n) (0x30 + (0x28 * n))
- #define **GPIO_INTSTAT**(n) (0x34 + (0x28 * n))
- #define **GPIO_REVID_REV** (0xFFFFFFFFu)
- #define **GPIO_REVID_REV_SHIFT** (0x00000000u)
- #define **GPIO_BINTEN_EN7** (0x00000080u)
- #define **GPIO_BINTEN_EN7_SHIFT** (0x00000007u)
- #define **GPIO_BINTEN_EN6** (0x00000040u)
- #define **GPIO_BINTEN_EN6_SHIFT** (0x00000006u)
- #define **GPIO_BINTEN_EN5** (0x00000020u)
- #define **GPIO_BINTEN_EN5_SHIFT** (0x00000005u)
- #define **GPIO_BINTEN_EN4** (0x00000010u)
- #define **GPIO_BINTEN_EN4_SHIFT** (0x00000004u)
- #define **GPIO_BINTEN_EN3** (0x00000008u)
- #define **GPIO_BINTEN_EN3_SHIFT** (0x00000003u)
- #define **GPIO_BINTEN_EN2** (0x00000004u)
- #define **GPIO_BINTEN_EN2_SHIFT** (0x00000002u)
- #define **GPIO_BINTEN_EN1** (0x00000002u)
- #define **GPIO_BINTEN_EN1_SHIFT** (0x00000001u)
- #define **GPIO_BINTEN_EN0** (0x00000001u)
- #define **GPIO_BINTEN_EN0_SHIFT** (0x00000000u)
- #define **GPIO_DIR_DIR31** (0x80000000u)
- #define **GPIO_DIR_DIR31_SHIFT** (0x0000001Fu)
- #define **GPIO_DIR_DIR30** (0x40000000u)
- #define **GPIO_DIR_DIR30_SHIFT** (0x0000001Eu)
- #define **GPIO_DIR_DIR29** (0x20000000u)
- #define **GPIO_DIR_DIR29_SHIFT** (0x0000001Du)
- #define **GPIO_DIR_DIR28** (0x10000000u)
- #define **GPIO_DIR_DIR28_SHIFT** (0x0000001Cu)
- #define **GPIO_DIR_DIR27** (0x08000000u)
- #define **GPIO_DIR_DIR27_SHIFT** (0x0000001Bu)
- #define **GPIO_DIR_DIR26** (0x04000000u)
- #define **GPIO_DIR_DIR26_SHIFT** (0x0000001Au)
- #define **GPIO_DIR_DIR25** (0x02000000u)

- #define **GPIO_DIR_DIR25_SHIFT** (0x00000019u)
- #define **GPIO_DIR_DIR24** (0x01000000u)
- #define **GPIO_DIR_DIR24_SHIFT** (0x00000018u)
- #define **GPIO_DIR_DIR23** (0x00800000u)
- #define **GPIO_DIR_DIR23_SHIFT** (0x00000017u)
- #define **GPIO_DIR_DIR22** (0x00400000u)
- #define **GPIO_DIR_DIR22_SHIFT** (0x00000016u)
- #define **GPIO_DIR_DIR21** (0x00200000u)
- #define **GPIO_DIR_DIR21_SHIFT** (0x00000015u)
- #define **GPIO_DIR_DIR20** (0x00100000u)
- #define **GPIO_DIR_DIR20_SHIFT** (0x00000014u)
- #define **GPIO_DIR_DIR19** (0x00080000u)
- #define **GPIO_DIR_DIR19_SHIFT** (0x00000013u)
- #define **GPIO_DIR_DIR18** (0x00040000u)
- #define **GPIO_DIR_DIR18_SHIFT** (0x00000012u)
- #define **GPIO_DIR_DIR17** (0x00020000u)
- #define **GPIO_DIR_DIR17_SHIFT** (0x00000011u)
- #define **GPIO_DIR_DIR16** (0x00010000u)
- #define **GPIO_DIR_DIR16_SHIFT** (0x00000010u)
- #define **GPIO_DIR_DIR15** (0x00008000u)
- #define **GPIO_DIR_DIR15_SHIFT** (0x0000000Fu)
- #define **GPIO_DIR_DIR14** (0x00004000u)
- #define **GPIO_DIR_DIR14_SHIFT** (0x0000000Eu)
- #define **GPIO_DIR_DIR13** (0x00002000u)
- #define **GPIO_DIR_DIR13_SHIFT** (0x0000000Du)
- #define **GPIO_DIR_DIR12** (0x00001000u)
- #define **GPIO_DIR_DIR12_SHIFT** (0x0000000Cu)
- #define **GPIO_DIR_DIR11** (0x00000800u)
- #define **GPIO_DIR_DIR11_SHIFT** (0x0000000Bu)
- #define **GPIO_DIR_DIR10** (0x00000400u)
- #define **GPIO_DIR_DIR10_SHIFT** (0x0000000Au)
- #define **GPIO_DIR_DIR9** (0x00000200u)
- #define **GPIO_DIR_DIR9_SHIFT** (0x00000009u)
- #define **GPIO_DIR_DIR8** (0x00000100u)
- #define **GPIO_DIR_DIR8_SHIFT** (0x00000008u)
- #define **GPIO_DIR_DIR7** (0x00000080u)
- #define **GPIO_DIR_DIR7_SHIFT** (0x00000007u)
- #define **GPIO_DIR_DIR6** (0x00000040u)
- #define **GPIO_DIR_DIR6_SHIFT** (0x00000006u)
- #define **GPIO_DIR_DIR5** (0x00000020u)
- #define **GPIO_DIR_DIR5_SHIFT** (0x00000005u)
- #define **GPIO_DIR_DIR4** (0x00000010u)
- #define **GPIO_DIR_DIR4_SHIFT** (0x00000004u)
- #define **GPIO_DIR_DIR3** (0x00000008u)
- #define **GPIO_DIR_DIR3_SHIFT** (0x00000003u)
- #define **GPIO_DIR_DIR2** (0x00000004u)
- #define **GPIO_DIR_DIR2_SHIFT** (0x00000002u)
- #define **GPIO_DIR_DIR1** (0x00000002u)
- #define **GPIO_DIR_DIR1_SHIFT** (0x00000001u)
- #define **GPIO_DIR_DIR0** (0x00000001u)
- #define **GPIO_DIR_DIR0_SHIFT** (0x00000000u)
- #define **GPIO_OUT_DATA_OUT31** (0x80000000u)
- #define **GPIO_OUT_DATA_OUT31_SHIFT** (0x0000001Fu)
- #define **GPIO_OUT_DATA_OUT30** (0x40000000u)
- #define **GPIO_OUT_DATA_OUT30_SHIFT** (0x0000001Eu)

- #define **GPIO_OUT_DATA_OUT29** (0x20000000u)
- #define **GPIO_OUT_DATA_OUT29_SHIFT** (0x0000001Du)
- #define **GPIO_OUT_DATA_OUT28** (0x10000000u)
- #define **GPIO_OUT_DATA_OUT28_SHIFT** (0x0000001Cu)
- #define **GPIO_OUT_DATA_OUT27** (0x08000000u)
- #define **GPIO_OUT_DATA_OUT27_SHIFT** (0x0000001Bu)
- #define **GPIO_OUT_DATA_OUT26** (0x04000000u)
- #define **GPIO_OUT_DATA_OUT26_SHIFT** (0x0000001Au)
- #define **GPIO_OUT_DATA_OUT25** (0x02000000u)
- #define **GPIO_OUT_DATA_OUT25_SHIFT** (0x00000019u)
- #define **GPIO_OUT_DATA_OUT24** (0x01000000u)
- #define **GPIO_OUT_DATA_OUT24_SHIFT** (0x00000018u)
- #define **GPIO_OUT_DATA_OUT23** (0x00800000u)
- #define **GPIO_OUT_DATA_OUT23_SHIFT** (0x00000017u)
- #define **GPIO_OUT_DATA_OUT22** (0x00400000u)
- #define **GPIO_OUT_DATA_OUT22_SHIFT** (0x00000016u)
- #define **GPIO_OUT_DATA_OUT21** (0x00200000u)
- #define **GPIO_OUT_DATA_OUT21_SHIFT** (0x00000015u)
- #define **GPIO_OUT_DATA_OUT20** (0x00100000u)
- #define **GPIO_OUT_DATA_OUT20_SHIFT** (0x00000014u)
- #define **GPIO_OUT_DATA_OUT19** (0x00080000u)
- #define **GPIO_OUT_DATA_OUT19_SHIFT** (0x00000013u)
- #define **GPIO_OUT_DATA_OUT18** (0x00040000u)
- #define **GPIO_OUT_DATA_OUT18_SHIFT** (0x00000012u)
- #define **GPIO_OUT_DATA_OUT17** (0x00020000u)
- #define **GPIO_OUT_DATA_OUT17_SHIFT** (0x00000011u)
- #define **GPIO_OUT_DATA_OUT16** (0x00010000u)
- #define **GPIO_OUT_DATA_OUT16_SHIFT** (0x00000010u)
- #define **GPIO_OUT_DATA_OUT15** (0x00008000u)
- #define **GPIO_OUT_DATA_OUT15_SHIFT** (0x0000000Fu)
- #define **GPIO_OUT_DATA_OUT14** (0x00004000u)
- #define **GPIO_OUT_DATA_OUT14_SHIFT** (0x0000000Eu)
- #define **GPIO_OUT_DATA_OUT13** (0x00002000u)
- #define **GPIO_OUT_DATA_OUT13_SHIFT** (0x0000000Du)
- #define **GPIO_OUT_DATA_OUT12** (0x00001000u)
- #define **GPIO_OUT_DATA_OUT12_SHIFT** (0x0000000Cu)
- #define **GPIO_OUT_DATA_OUT11** (0x00000800u)
- #define **GPIO_OUT_DATA_OUT11_SHIFT** (0x0000000Bu)
- #define **GPIO_OUT_DATA_OUT10** (0x00000400u)
- #define **GPIO_OUT_DATA_OUT10_SHIFT** (0x0000000Au)
- #define **GPIO_OUT_DATA_OUT9** (0x00000200u)
- #define **GPIO_OUT_DATA_OUT9_SHIFT** (0x00000009u)
- #define **GPIO_OUT_DATA_OUT8** (0x00000100u)
- #define **GPIO_OUT_DATA_OUT8_SHIFT** (0x00000008u)
- #define **GPIO_OUT_DATA_OUT7** (0x00000080u)
- #define **GPIO_OUT_DATA_OUT7_SHIFT** (0x00000007u)
- #define **GPIO_OUT_DATA_OUT6** (0x00000040u)
- #define **GPIO_OUT_DATA_OUT6_SHIFT** (0x00000006u)
- #define **GPIO_OUT_DATA_OUT5** (0x00000020u)
- #define **GPIO_OUT_DATA_OUT5_SHIFT** (0x00000005u)
- #define **GPIO_OUT_DATA_OUT4** (0x00000010u)
- #define **GPIO_OUT_DATA_OUT4_SHIFT** (0x00000004u)
- #define **GPIO_OUT_DATA_OUT3** (0x00000008u)
- #define **GPIO_OUT_DATA_OUT3_SHIFT** (0x00000003u)
- #define **GPIO_OUT_DATA_OUT2** (0x00000004u)

- #define **GPIO_OUT_DATA_OUT2_SHIFT** (0x00000002u)
- #define **GPIO_OUT_DATA_OUT1** (0x00000002u)
- #define **GPIO_OUT_DATA_OUT1_SHIFT** (0x00000001u)
- #define **GPIO_OUT_DATA_OUT0** (0x00000001u)
- #define **GPIO_OUT_DATA_OUT0_SHIFT** (0x00000000u)
- #define **GPIO_SET_DATA_SET31** (0x80000000u)
- #define **GPIO_SET_DATA_SET31_SHIFT** (0x0000001Fu)
- #define **GPIO_SET_DATA_SET30** (0x40000000u)
- #define **GPIO_SET_DATA_SET30_SHIFT** (0x0000001Eu)
- #define **GPIO_SET_DATA_SET29** (0x20000000u)
- #define **GPIO_SET_DATA_SET29_SHIFT** (0x0000001Du)
- #define **GPIO_SET_DATA_SET28** (0x10000000u)
- #define **GPIO_SET_DATA_SET28_SHIFT** (0x0000001Cu)
- #define **GPIO_SET_DATA_SET27** (0x08000000u)
- #define **GPIO_SET_DATA_SET27_SHIFT** (0x0000001Bu)
- #define **GPIO_SET_DATA_SET26** (0x04000000u)
- #define **GPIO_SET_DATA_SET26_SHIFT** (0x0000001Au)
- #define **GPIO_SET_DATA_SET25** (0x02000000u)
- #define **GPIO_SET_DATA_SET25_SHIFT** (0x00000019u)
- #define **GPIO_SET_DATA_SET24** (0x01000000u)
- #define **GPIO_SET_DATA_SET24_SHIFT** (0x00000018u)
- #define **GPIO_SET_DATA_SET23** (0x00800000u)
- #define **GPIO_SET_DATA_SET23_SHIFT** (0x00000017u)
- #define **GPIO_SET_DATA_SET22** (0x00400000u)
- #define **GPIO_SET_DATA_SET22_SHIFT** (0x00000016u)
- #define **GPIO_SET_DATA_SET21** (0x00200000u)
- #define **GPIO_SET_DATA_SET21_SHIFT** (0x00000015u)
- #define **GPIO_SET_DATA_SET20** (0x00100000u)
- #define **GPIO_SET_DATA_SET20_SHIFT** (0x00000014u)
- #define **GPIO_SET_DATA_SET19** (0x00080000u)
- #define **GPIO_SET_DATA_SET19_SHIFT** (0x00000013u)
- #define **GPIO_SET_DATA_SET18** (0x00040000u)
- #define **GPIO_SET_DATA_SET18_SHIFT** (0x00000012u)
- #define **GPIO_SET_DATA_SET17** (0x00020000u)
- #define **GPIO_SET_DATA_SET17_SHIFT** (0x00000011u)
- #define **GPIO_SET_DATA_SET16** (0x00010000u)
- #define **GPIO_SET_DATA_SET16_SHIFT** (0x00000010u)
- #define **GPIO_SET_DATA_SET15** (0x00008000u)
- #define **GPIO_SET_DATA_SET15_SHIFT** (0x0000000Fu)
- #define **GPIO_SET_DATA_SET14** (0x00004000u)
- #define **GPIO_SET_DATA_SET14_SHIFT** (0x0000000Eu)
- #define **GPIO_SET_DATA_SET13** (0x00002000u)
- #define **GPIO_SET_DATA_SET13_SHIFT** (0x0000000Du)
- #define **GPIO_SET_DATA_SET12** (0x00001000u)
- #define **GPIO_SET_DATA_SET12_SHIFT** (0x0000000Cu)
- #define **GPIO_SET_DATA_SET11** (0x00000800u)
- #define **GPIO_SET_DATA_SET11_SHIFT** (0x0000000Bu)
- #define **GPIO_SET_DATA_SET10** (0x00000400u)
- #define **GPIO_SET_DATA_SET10_SHIFT** (0x0000000Au)
- #define **GPIO_SET_DATA_SET9** (0x00000200u)
- #define **GPIO_SET_DATA_SET9_SHIFT** (0x00000009u)
- #define **GPIO_SET_DATA_SET8** (0x00000100u)
- #define **GPIO_SET_DATA_SET8_SHIFT** (0x00000008u)
- #define **GPIO_SET_DATA_SET7** (0x00000080u)
- #define **GPIO_SET_DATA_SET7_SHIFT** (0x00000007u)

- #define **GPIO_SET_DATA_SET6** (0x00000040u)
- #define **GPIO_SET_DATA_SET6_SHIFT** (0x00000006u)
- #define **GPIO_SET_DATA_SET5** (0x00000020u)
- #define **GPIO_SET_DATA_SET5_SHIFT** (0x00000005u)
- #define **GPIO_SET_DATA_SET4** (0x00000010u)
- #define **GPIO_SET_DATA_SET4_SHIFT** (0x00000004u)
- #define **GPIO_SET_DATA_SET3** (0x00000008u)
- #define **GPIO_SET_DATA_SET3_SHIFT** (0x00000003u)
- #define **GPIO_SET_DATA_SET2** (0x00000004u)
- #define **GPIO_SET_DATA_SET2_SHIFT** (0x00000002u)
- #define **GPIO_SET_DATA_SET1** (0x00000002u)
- #define **GPIO_SET_DATA_SET1_SHIFT** (0x00000001u)
- #define **GPIO_SET_DATA_SET0** (0x00000001u)
- #define **GPIO_SET_DATA_SET0_SHIFT** (0x00000000u)
- #define **GPIO_CLR_DATA_CLR31** (0x80000000u)
- #define **GPIO_CLR_DATA_CLR31_SHIFT** (0x0000001Fu)
- #define **GPIO_CLR_DATA_CLR30** (0x40000000u)
- #define **GPIO_CLR_DATA_CLR30_SHIFT** (0x0000001Eu)
- #define **GPIO_CLR_DATA_CLR29** (0x20000000u)
- #define **GPIO_CLR_DATA_CLR29_SHIFT** (0x0000001Du)
- #define **GPIO_CLR_DATA_CLR28** (0x10000000u)
- #define **GPIO_CLR_DATA_CLR28_SHIFT** (0x0000001Cu)
- #define **GPIO_CLR_DATA_CLR27** (0x08000000u)
- #define **GPIO_CLR_DATA_CLR27_SHIFT** (0x0000001Bu)
- #define **GPIO_CLR_DATA_CLR26** (0x04000000u)
- #define **GPIO_CLR_DATA_CLR26_SHIFT** (0x0000001Au)
- #define **GPIO_CLR_DATA_CLR25** (0x02000000u)
- #define **GPIO_CLR_DATA_CLR25_SHIFT** (0x00000019u)
- #define **GPIO_CLR_DATA_CLR24** (0x01000000u)
- #define **GPIO_CLR_DATA_CLR24_SHIFT** (0x00000018u)
- #define **GPIO_CLR_DATA_CLR23** (0x00800000u)
- #define **GPIO_CLR_DATA_CLR23_SHIFT** (0x00000017u)
- #define **GPIO_CLR_DATA_CLR22** (0x00400000u)
- #define **GPIO_CLR_DATA_CLR22_SHIFT** (0x00000016u)
- #define **GPIO_CLR_DATA_CLR21** (0x00200000u)
- #define **GPIO_CLR_DATA_CLR21_SHIFT** (0x00000015u)
- #define **GPIO_CLR_DATA_CLR20** (0x00100000u)
- #define **GPIO_CLR_DATA_CLR20_SHIFT** (0x00000014u)
- #define **GPIO_CLR_DATA_CLR19** (0x00080000u)
- #define **GPIO_CLR_DATA_CLR19_SHIFT** (0x00000013u)
- #define **GPIO_CLR_DATA_CLR18** (0x00040000u)
- #define **GPIO_CLR_DATA_CLR18_SHIFT** (0x00000012u)
- #define **GPIO_CLR_DATA_CLR17** (0x00020000u)
- #define **GPIO_CLR_DATA_CLR17_SHIFT** (0x00000011u)
- #define **GPIO_CLR_DATA_CLR16** (0x00010000u)
- #define **GPIO_CLR_DATA_CLR16_SHIFT** (0x00000010u)
- #define **GPIO_CLR_DATA_CLR15** (0x00008000u)
- #define **GPIO_CLR_DATA_CLR15_SHIFT** (0x0000000Fu)
- #define **GPIO_CLR_DATA_CLR14** (0x00004000u)
- #define **GPIO_CLR_DATA_CLR14_SHIFT** (0x0000000Eu)
- #define **GPIO_CLR_DATA_CLR13** (0x00002000u)
- #define **GPIO_CLR_DATA_CLR13_SHIFT** (0x0000000Du)
- #define **GPIO_CLR_DATA_CLR12** (0x00001000u)
- #define **GPIO_CLR_DATA_CLR12_SHIFT** (0x0000000Cu)
- #define **GPIO_CLR_DATA_CLR11** (0x00000800u)

- #define **GPIO_CLR_DATA_CLR11_SHIFT** (0x0000000Bu)
- #define **GPIO_CLR_DATA_CLR10** (0x00000400u)
- #define **GPIO_CLR_DATA_CLR10_SHIFT** (0x0000000Au)
- #define **GPIO_CLR_DATA_CLR9** (0x00000200u)
- #define **GPIO_CLR_DATA_CLR9_SHIFT** (0x00000009u)
- #define **GPIO_CLR_DATA_CLR8** (0x00000100u)
- #define **GPIO_CLR_DATA_CLR8_SHIFT** (0x00000008u)
- #define **GPIO_CLR_DATA_CLR7** (0x00000080u)
- #define **GPIO_CLR_DATA_CLR7_SHIFT** (0x00000007u)
- #define **GPIO_CLR_DATA_CLR6** (0x00000040u)
- #define **GPIO_CLR_DATA_CLR6_SHIFT** (0x00000006u)
- #define **GPIO_CLR_DATA_CLR5** (0x00000020u)
- #define **GPIO_CLR_DATA_CLR5_SHIFT** (0x00000005u)
- #define **GPIO_CLR_DATA_CLR4** (0x00000010u)
- #define **GPIO_CLR_DATA_CLR4_SHIFT** (0x00000004u)
- #define **GPIO_CLR_DATA_CLR3** (0x00000008u)
- #define **GPIO_CLR_DATA_CLR3_SHIFT** (0x00000003u)
- #define **GPIO_CLR_DATA_CLR2** (0x00000004u)
- #define **GPIO_CLR_DATA_CLR2_SHIFT** (0x00000002u)
- #define **GPIO_CLR_DATA_CLR1** (0x00000002u)
- #define **GPIO_CLR_DATA_CLR1_SHIFT** (0x00000001u)
- #define **GPIO_CLR_DATA_CLR0** (0x00000001u)
- #define **GPIO_CLR_DATA_CLR0_SHIFT** (0x00000000u)
- #define **GPIO_IN_DATA_IN31** (0x80000000u)
- #define **GPIO_IN_DATA_IN31_SHIFT** (0x0000001Fu)
- #define **GPIO_IN_DATA_IN30** (0x40000000u)
- #define **GPIO_IN_DATA_IN30_SHIFT** (0x0000001Eu)
- #define **GPIO_IN_DATA_IN29** (0x20000000u)
- #define **GPIO_IN_DATA_IN29_SHIFT** (0x0000001Du)
- #define **GPIO_IN_DATA_IN28** (0x10000000u)
- #define **GPIO_IN_DATA_IN28_SHIFT** (0x0000001Cu)
- #define **GPIO_IN_DATA_IN27** (0x08000000u)
- #define **GPIO_IN_DATA_IN27_SHIFT** (0x0000001Bu)
- #define **GPIO_IN_DATA_IN26** (0x04000000u)
- #define **GPIO_IN_DATA_IN26_SHIFT** (0x0000001Au)
- #define **GPIO_IN_DATA_IN25** (0x02000000u)
- #define **GPIO_IN_DATA_IN25_SHIFT** (0x00000019u)
- #define **GPIO_IN_DATA_IN24** (0x01000000u)
- #define **GPIO_IN_DATA_IN24_SHIFT** (0x00000018u)
- #define **GPIO_IN_DATA_IN23** (0x00800000u)
- #define **GPIO_IN_DATA_IN23_SHIFT** (0x00000017u)
- #define **GPIO_IN_DATA_IN22** (0x00400000u)
- #define **GPIO_IN_DATA_IN22_SHIFT** (0x00000016u)
- #define **GPIO_IN_DATA_IN21** (0x00200000u)
- #define **GPIO_IN_DATA_IN21_SHIFT** (0x00000015u)
- #define **GPIO_IN_DATA_IN20** (0x00100000u)
- #define **GPIO_IN_DATA_IN20_SHIFT** (0x00000014u)
- #define **GPIO_IN_DATA_IN19** (0x00080000u)
- #define **GPIO_IN_DATA_IN19_SHIFT** (0x00000013u)
- #define **GPIO_IN_DATA_IN18** (0x00040000u)
- #define **GPIO_IN_DATA_IN18_SHIFT** (0x00000012u)
- #define **GPIO_IN_DATA_IN17** (0x00020000u)
- #define **GPIO_IN_DATA_IN17_SHIFT** (0x00000011u)
- #define **GPIO_IN_DATA_IN16** (0x00010000u)
- #define **GPIO_IN_DATA_IN16_SHIFT** (0x00000010u)

- #define **GPIO_IN_DATA_IN15** (0x00008000u)
- #define **GPIO_IN_DATA_IN15_SHIFT** (0x0000000Fu)
- #define **GPIO_IN_DATA_IN14** (0x00004000u)
- #define **GPIO_IN_DATA_IN14_SHIFT** (0x0000000Eu)
- #define **GPIO_IN_DATA_IN13** (0x00002000u)
- #define **GPIO_IN_DATA_IN13_SHIFT** (0x0000000Du)
- #define **GPIO_IN_DATA_IN12** (0x00001000u)
- #define **GPIO_IN_DATA_IN12_SHIFT** (0x0000000Cu)
- #define **GPIO_IN_DATA_IN11** (0x00000800u)
- #define **GPIO_IN_DATA_IN11_SHIFT** (0x0000000Bu)
- #define **GPIO_IN_DATA_IN10** (0x00000400u)
- #define **GPIO_IN_DATA_IN10_SHIFT** (0x0000000Au)
- #define **GPIO_IN_DATA_IN9** (0x00000200u)
- #define **GPIO_IN_DATA_IN9_SHIFT** (0x00000009u)
- #define **GPIO_IN_DATA_IN8** (0x00000100u)
- #define **GPIO_IN_DATA_IN8_SHIFT** (0x00000008u)
- #define **GPIO_IN_DATA_IN7** (0x00000080u)
- #define **GPIO_IN_DATA_IN7_SHIFT** (0x00000007u)
- #define **GPIO_IN_DATA_IN6** (0x00000040u)
- #define **GPIO_IN_DATA_IN6_SHIFT** (0x00000006u)
- #define **GPIO_IN_DATA_IN5** (0x00000020u)
- #define **GPIO_IN_DATA_IN5_SHIFT** (0x00000005u)
- #define **GPIO_IN_DATA_IN4** (0x00000010u)
- #define **GPIO_IN_DATA_IN4_SHIFT** (0x00000004u)
- #define **GPIO_IN_DATA_IN3** (0x00000008u)
- #define **GPIO_IN_DATA_IN3_SHIFT** (0x00000003u)
- #define **GPIO_IN_DATA_IN2** (0x00000004u)
- #define **GPIO_IN_DATA_IN2_SHIFT** (0x00000002u)
- #define **GPIO_IN_DATA_IN1** (0x00000002u)
- #define **GPIO_IN_DATA_IN1_SHIFT** (0x00000001u)
- #define **GPIO_IN_DATA_IN0** (0x00000001u)
- #define **GPIO_IN_DATA_IN0_SHIFT** (0x00000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS31** (0x80000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS31_SHIFT** (0x0000001Fu)
- #define **GPIO_SET_RIS_TRIG_SETRIS30** (0x40000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS30_SHIFT** (0x0000001Eu)
- #define **GPIO_SET_RIS_TRIG_SETRIS29** (0x20000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS29_SHIFT** (0x0000001Du)
- #define **GPIO_SET_RIS_TRIG_SETRIS28** (0x10000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS28_SHIFT** (0x0000001Cu)
- #define **GPIO_SET_RIS_TRIG_SETRIS27** (0x08000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS27_SHIFT** (0x0000001Bu)
- #define **GPIO_SET_RIS_TRIG_SETRIS26** (0x04000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS26_SHIFT** (0x0000001Au)
- #define **GPIO_SET_RIS_TRIG_SETRIS25** (0x02000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS25_SHIFT** (0x00000019u)
- #define **GPIO_SET_RIS_TRIG_SETRIS24** (0x01000000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS24_SHIFT** (0x00000018u)
- #define **GPIO_SET_RIS_TRIG_SETRIS23** (0x00800000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS23_SHIFT** (0x00000017u)
- #define **GPIO_SET_RIS_TRIG_SETRIS22** (0x00400000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS22_SHIFT** (0x00000016u)
- #define **GPIO_SET_RIS_TRIG_SETRIS21** (0x00200000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS21_SHIFT** (0x00000015u)
- #define **GPIO_SET_RIS_TRIG_SETRIS20** (0x00100000u)

- #define **GPIO_SET_RIS_TRIG_SETRIS20_SHIFT** (0x00000014u)
- #define **GPIO_SET_RIS_TRIG_SETRIS19** (0x00080000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS19_SHIFT** (0x00000013u)
- #define **GPIO_SET_RIS_TRIG_SETRIS18** (0x00040000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS18_SHIFT** (0x00000012u)
- #define **GPIO_SET_RIS_TRIG_SETRIS17** (0x00020000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS17_SHIFT** (0x00000011u)
- #define **GPIO_SET_RIS_TRIG_SETRIS16** (0x00010000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS16_SHIFT** (0x00000010u)
- #define **GPIO_SET_RIS_TRIG_SETRIS15** (0x00008000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS15_SHIFT** (0x0000000Fu)
- #define **GPIO_SET_RIS_TRIG_SETRIS14** (0x00004000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS14_SHIFT** (0x0000000Eu)
- #define **GPIO_SET_RIS_TRIG_SETRIS13** (0x00002000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS13_SHIFT** (0x0000000Du)
- #define **GPIO_SET_RIS_TRIG_SETRIS12** (0x00001000u)
- #define **GPIO_SET_RIS_TRIG_SETRIS12_SHIFT** (0x0000000Cu)
- #define **GPIO_SET_RIS_TRIG_SETRIS11** (0x00000800u)
- #define **GPIO_SET_RIS_TRIG_SETRIS11_SHIFT** (0x0000000Bu)
- #define **GPIO_SET_RIS_TRIG_SETRIS10** (0x00000400u)
- #define **GPIO_SET_RIS_TRIG_SETRIS10_SHIFT** (0x0000000Au)
- #define **GPIO_SET_RIS_TRIG_SETRIS9** (0x00000200u)
- #define **GPIO_SET_RIS_TRIG_SETRIS9_SHIFT** (0x00000009u)
- #define **GPIO_SET_RIS_TRIG_SETRIS8** (0x00000100u)
- #define **GPIO_SET_RIS_TRIG_SETRIS8_SHIFT** (0x00000008u)
- #define **GPIO_SET_RIS_TRIG_SETRIS7** (0x00000080u)
- #define **GPIO_SET_RIS_TRIG_SETRIS7_SHIFT** (0x00000007u)
- #define **GPIO_SET_RIS_TRIG_SETRIS6** (0x00000040u)
- #define **GPIO_SET_RIS_TRIG_SETRIS6_SHIFT** (0x00000006u)
- #define **GPIO_SET_RIS_TRIG_SETRIS5** (0x00000020u)
- #define **GPIO_SET_RIS_TRIG_SETRIS5_SHIFT** (0x00000005u)
- #define **GPIO_SET_RIS_TRIG_SETRIS4** (0x00000010u)
- #define **GPIO_SET_RIS_TRIG_SETRIS4_SHIFT** (0x00000004u)
- #define **GPIO_SET_RIS_TRIG_SETRIS3** (0x00000008u)
- #define **GPIO_SET_RIS_TRIG_SETRIS3_SHIFT** (0x00000003u)
- #define **GPIO_SET_RIS_TRIG_SETRIS2** (0x00000004u)
- #define **GPIO_SET_RIS_TRIG_SETRIS2_SHIFT** (0x00000002u)
- #define **GPIO_SET_RIS_TRIG_SETRIS1** (0x00000002u)
- #define **GPIO_SET_RIS_TRIG_SETRIS1_SHIFT** (0x00000001u)
- #define **GPIO_SET_RIS_TRIG_SETRIS0** (0x00000001u)
- #define **GPIO_SET_RIS_TRIG_SETRIS0_SHIFT** (0x00000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS31** (0x80000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS31_SHIFT** (0x0000001Fu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS30** (0x40000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS30_SHIFT** (0x0000001Eu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS29** (0x20000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS29_SHIFT** (0x0000001Du)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS28** (0x10000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS28_SHIFT** (0x0000001Cu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS27** (0x08000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS27_SHIFT** (0x0000001Bu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS26** (0x04000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS26_SHIFT** (0x0000001Au)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS25** (0x02000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS25_SHIFT** (0x00000019u)

- #define **GPIO_CLR_RIS_TRIG_CLRRIS24** (0x01000000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS24_SHIFT** (0x00000018u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS23** (0x00800000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS23_SHIFT** (0x00000017u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS22** (0x00400000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS22_SHIFT** (0x00000016u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS21** (0x00200000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS21_SHIFT** (0x00000015u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS20** (0x00100000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS20_SHIFT** (0x00000014u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS19** (0x00080000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS19_SHIFT** (0x00000013u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS18** (0x00040000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS18_SHIFT** (0x00000012u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS17** (0x00020000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS17_SHIFT** (0x00000011u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS16** (0x00010000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS16_SHIFT** (0x00000010u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS15** (0x00008000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS15_SHIFT** (0x0000000Fu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS14** (0x00004000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS14_SHIFT** (0x0000000Eu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS13** (0x00002000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS13_SHIFT** (0x0000000Du)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS12** (0x00001000u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS12_SHIFT** (0x0000000Cu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS11** (0x00000800u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS11_SHIFT** (0x0000000Bu)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS10** (0x00000400u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS10_SHIFT** (0x0000000Au)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS9** (0x00000200u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS9_SHIFT** (0x00000009u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS8** (0x00000100u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS8_SHIFT** (0x00000008u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS7** (0x00000080u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS7_SHIFT** (0x00000007u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS6** (0x00000040u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS6_SHIFT** (0x00000006u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS5** (0x00000020u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS5_SHIFT** (0x00000005u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS4** (0x00000010u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS4_SHIFT** (0x00000004u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS3** (0x00000008u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS3_SHIFT** (0x00000003u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS2** (0x00000004u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS2_SHIFT** (0x00000002u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS1** (0x00000002u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS1_SHIFT** (0x00000001u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS0** (0x00000001u)
- #define **GPIO_CLR_RIS_TRIG_CLRRIS0_SHIFT** (0x00000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL31** (0x80000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL31_SHIFT** (0x0000001Fu)
- #define **GPIO_SET_FAL_TRIG_SETFAL30** (0x40000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL30_SHIFT** (0x0000001Eu)
- #define **GPIO_SET_FAL_TRIG_SETFAL29** (0x20000000u)

- #define **GPIO_SET_FAL_TRIG_SETFAL29_SHIFT** (0x0000001Du)
- #define **GPIO_SET_FAL_TRIG_SETFAL28** (0x10000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL28_SHIFT** (0x0000001Cu)
- #define **GPIO_SET_FAL_TRIG_SETFAL27** (0x08000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL27_SHIFT** (0x0000001Bu)
- #define **GPIO_SET_FAL_TRIG_SETFAL26** (0x04000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL26_SHIFT** (0x0000001Au)
- #define **GPIO_SET_FAL_TRIG_SETFAL25** (0x02000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL25_SHIFT** (0x00000019u)
- #define **GPIO_SET_FAL_TRIG_SETFAL24** (0x01000000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL24_SHIFT** (0x00000018u)
- #define **GPIO_SET_FAL_TRIG_SETFAL23** (0x00800000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL23_SHIFT** (0x00000017u)
- #define **GPIO_SET_FAL_TRIG_SETFAL22** (0x00400000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL22_SHIFT** (0x00000016u)
- #define **GPIO_SET_FAL_TRIG_SETFAL21** (0x00200000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL21_SHIFT** (0x00000015u)
- #define **GPIO_SET_FAL_TRIG_SETFAL20** (0x00100000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL20_SHIFT** (0x00000014u)
- #define **GPIO_SET_FAL_TRIG_SETFAL19** (0x00080000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL19_SHIFT** (0x00000013u)
- #define **GPIO_SET_FAL_TRIG_SETFAL18** (0x00040000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL18_SHIFT** (0x00000012u)
- #define **GPIO_SET_FAL_TRIG_SETFAL17** (0x00020000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL17_SHIFT** (0x00000011u)
- #define **GPIO_SET_FAL_TRIG_SETFAL16** (0x00010000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL16_SHIFT** (0x00000010u)
- #define **GPIO_SET_FAL_TRIG_SETFAL15** (0x00008000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL15_SHIFT** (0x0000000Fu)
- #define **GPIO_SET_FAL_TRIG_SETFAL14** (0x00004000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL14_SHIFT** (0x0000000Eu)
- #define **GPIO_SET_FAL_TRIG_SETFAL13** (0x00002000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL13_SHIFT** (0x0000000Du)
- #define **GPIO_SET_FAL_TRIG_SETFAL12** (0x00001000u)
- #define **GPIO_SET_FAL_TRIG_SETFAL12_SHIFT** (0x0000000Cu)
- #define **GPIO_SET_FAL_TRIG_SETFAL11** (0x00000800u)
- #define **GPIO_SET_FAL_TRIG_SETFAL11_SHIFT** (0x0000000Bu)
- #define **GPIO_SET_FAL_TRIG_SETFAL10** (0x00000400u)
- #define **GPIO_SET_FAL_TRIG_SETFAL10_SHIFT** (0x0000000Au)
- #define **GPIO_SET_FAL_TRIG_SETFAL9** (0x00000200u)
- #define **GPIO_SET_FAL_TRIG_SETFAL9_SHIFT** (0x00000009u)
- #define **GPIO_SET_FAL_TRIG_SETFAL8** (0x00000100u)
- #define **GPIO_SET_FAL_TRIG_SETFAL8_SHIFT** (0x00000008u)
- #define **GPIO_SET_FAL_TRIG_SETFAL7** (0x00000080u)
- #define **GPIO_SET_FAL_TRIG_SETFAL7_SHIFT** (0x00000007u)
- #define **GPIO_SET_FAL_TRIG_SETFAL6** (0x00000040u)
- #define **GPIO_SET_FAL_TRIG_SETFAL6_SHIFT** (0x00000006u)
- #define **GPIO_SET_FAL_TRIG_SETFAL5** (0x00000020u)
- #define **GPIO_SET_FAL_TRIG_SETFAL5_SHIFT** (0x00000005u)
- #define **GPIO_SET_FAL_TRIG_SETFAL4** (0x00000010u)
- #define **GPIO_SET_FAL_TRIG_SETFAL4_SHIFT** (0x00000004u)
- #define **GPIO_SET_FAL_TRIG_SETFAL3** (0x00000008u)
- #define **GPIO_SET_FAL_TRIG_SETFAL3_SHIFT** (0x00000003u)
- #define **GPIO_SET_FAL_TRIG_SETFAL2** (0x00000004u)
- #define **GPIO_SET_FAL_TRIG_SETFAL2_SHIFT** (0x00000002u)

- #define **GPIO_SET_FAL_TRIG_SETFAL1** (0x00000002u)
- #define **GPIO_SET_FAL_TRIG_SETFAL1_SHIFT** (0x00000001u)
- #define **GPIO_SET_FAL_TRIG_SETFAL0** (0x00000001u)
- #define **GPIO_SET_FAL_TRIG_SETFAL0_SHIFT** (0x00000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL31** (0x80000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL31_SHIFT** (0x0000001Fu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL30** (0x40000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL30_SHIFT** (0x0000001Eu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL29** (0x20000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL29_SHIFT** (0x0000001Du)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL28** (0x10000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL28_SHIFT** (0x0000001Cu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL27** (0x08000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL27_SHIFT** (0x0000001Bu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL26** (0x04000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL26_SHIFT** (0x0000001Au)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL25** (0x02000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL25_SHIFT** (0x00000019u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL24** (0x01000000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL24_SHIFT** (0x00000018u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL23** (0x00800000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL23_SHIFT** (0x00000017u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL22** (0x00400000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL22_SHIFT** (0x00000016u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL21** (0x00200000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL21_SHIFT** (0x00000015u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL20** (0x00100000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL20_SHIFT** (0x00000014u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL19** (0x00080000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL19_SHIFT** (0x00000013u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL18** (0x00040000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL18_SHIFT** (0x00000012u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL17** (0x00020000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL17_SHIFT** (0x00000011u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL16** (0x00010000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL16_SHIFT** (0x00000010u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL15** (0x00008000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL15_SHIFT** (0x0000000Fu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL14** (0x00004000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL14_SHIFT** (0x0000000Eu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL13** (0x00002000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL13_SHIFT** (0x0000000Du)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL12** (0x00001000u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL12_SHIFT** (0x0000000Cu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL11** (0x00000800u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL11_SHIFT** (0x0000000Bu)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL10** (0x00000400u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL10_SHIFT** (0x0000000Au)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL9** (0x00000200u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL9_SHIFT** (0x00000009u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL8** (0x00000100u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL8_SHIFT** (0x00000008u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL7** (0x00000080u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL7_SHIFT** (0x00000007u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL6** (0x00000040u)

- #define **GPIO_CLR_FAL_TRIG_CLRFAL6_SHIFT** (0x00000006u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL5** (0x00000020u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL5_SHIFT** (0x00000005u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL4** (0x00000010u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL4_SHIFT** (0x00000004u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL3** (0x00000008u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL3_SHIFT** (0x00000003u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL2** (0x00000004u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL2_SHIFT** (0x00000002u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL1** (0x00000002u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL1_SHIFT** (0x00000001u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL0** (0x00000001u)
- #define **GPIO_CLR_FAL_TRIG_CLRFAL0_SHIFT** (0x00000000u)
- #define **GPIO_INTSTAT_STAT31** (0x80000000u)
- #define **GPIO_INTSTAT_STAT31_SHIFT** (0x0000001Fu)
- #define **GPIO_INTSTAT_STAT30** (0x40000000u)
- #define **GPIO_INTSTAT_STAT30_SHIFT** (0x0000001Eu)
- #define **GPIO_INTSTAT_STAT29** (0x20000000u)
- #define **GPIO_INTSTAT_STAT29_SHIFT** (0x0000001Du)
- #define **GPIO_INTSTAT_STAT28** (0x10000000u)
- #define **GPIO_INTSTAT_STAT28_SHIFT** (0x0000001Cu)
- #define **GPIO_INTSTAT_STAT27** (0x08000000u)
- #define **GPIO_INTSTAT_STAT27_SHIFT** (0x0000001Bu)
- #define **GPIO_INTSTAT_STAT26** (0x04000000u)
- #define **GPIO_INTSTAT_STAT26_SHIFT** (0x0000001Au)
- #define **GPIO_INTSTAT_STAT25** (0x02000000u)
- #define **GPIO_INTSTAT_STAT25_SHIFT** (0x00000019u)
- #define **GPIO_INTSTAT_STAT24** (0x01000000u)
- #define **GPIO_INTSTAT_STAT24_SHIFT** (0x00000018u)
- #define **GPIO_INTSTAT_STAT23** (0x00800000u)
- #define **GPIO_INTSTAT_STAT23_SHIFT** (0x00000017u)
- #define **GPIO_INTSTAT_STAT22** (0x00400000u)
- #define **GPIO_INTSTAT_STAT22_SHIFT** (0x00000016u)
- #define **GPIO_INTSTAT_STAT21** (0x00200000u)
- #define **GPIO_INTSTAT_STAT21_SHIFT** (0x00000015u)
- #define **GPIO_INTSTAT_STAT20** (0x00100000u)
- #define **GPIO_INTSTAT_STAT20_SHIFT** (0x00000014u)
- #define **GPIO_INTSTAT_STAT19** (0x00080000u)
- #define **GPIO_INTSTAT_STAT19_SHIFT** (0x00000013u)
- #define **GPIO_INTSTAT_STAT18** (0x00040000u)
- #define **GPIO_INTSTAT_STAT18_SHIFT** (0x00000012u)
- #define **GPIO_INTSTAT_STAT17** (0x00020000u)
- #define **GPIO_INTSTAT_STAT17_SHIFT** (0x00000011u)
- #define **GPIO_INTSTAT_STAT16** (0x00010000u)
- #define **GPIO_INTSTAT_STAT16_SHIFT** (0x00000010u)
- #define **GPIO_INTSTAT_STAT15** (0x00008000u)
- #define **GPIO_INTSTAT_STAT15_SHIFT** (0x0000000Fu)
- #define **GPIO_INTSTAT_STAT14** (0x00004000u)
- #define **GPIO_INTSTAT_STAT14_SHIFT** (0x0000000Eu)
- #define **GPIO_INTSTAT_STAT13** (0x00002000u)
- #define **GPIO_INTSTAT_STAT13_SHIFT** (0x0000000Du)
- #define **GPIO_INTSTAT_STAT12** (0x00001000u)
- #define **GPIO_INTSTAT_STAT12_SHIFT** (0x0000000Cu)
- #define **GPIO_INTSTAT_STAT11** (0x00000800u)
- #define **GPIO_INTSTAT_STAT11_SHIFT** (0x0000000Bu)

- #define **GPIO_INTSTAT_STAT10** (0x00000400u)
- #define **GPIO_INTSTAT_STAT10_SHIFT** (0x0000000Au)
- #define **GPIO_INTSTAT_STAT9** (0x00000200u)
- #define **GPIO_INTSTAT_STAT9_SHIFT** (0x00000009u)
- #define **GPIO_INTSTAT_STAT8** (0x00000100u)
- #define **GPIO_INTSTAT_STAT8_SHIFT** (0x00000008u)
- #define **GPIO_INTSTAT_STAT7** (0x00000080u)
- #define **GPIO_INTSTAT_STAT7_SHIFT** (0x00000007u)
- #define **GPIO_INTSTAT_STAT6** (0x00000040u)
- #define **GPIO_INTSTAT_STAT6_SHIFT** (0x00000006u)
- #define **GPIO_INTSTAT_STAT5** (0x00000020u)
- #define **GPIO_INTSTAT_STAT5_SHIFT** (0x00000005u)
- #define **GPIO_INTSTAT_STAT4** (0x00000010u)
- #define **GPIO_INTSTAT_STAT4_SHIFT** (0x00000004u)
- #define **GPIO_INTSTAT_STAT3** (0x00000008u)
- #define **GPIO_INTSTAT_STAT3_SHIFT** (0x00000003u)
- #define **GPIO_INTSTAT_STAT2** (0x00000004u)
- #define **GPIO_INTSTAT_STAT2_SHIFT** (0x00000002u)
- #define **GPIO_INTSTAT_STAT1** (0x00000002u)
- #define **GPIO_INTSTAT_STAT1_SHIFT** (0x00000001u)
- #define **GPIO_INTSTAT_STAT0** (0x00000001u)
- #define **GPIO_INTSTAT_STAT0_SHIFT** (0x00000000u)

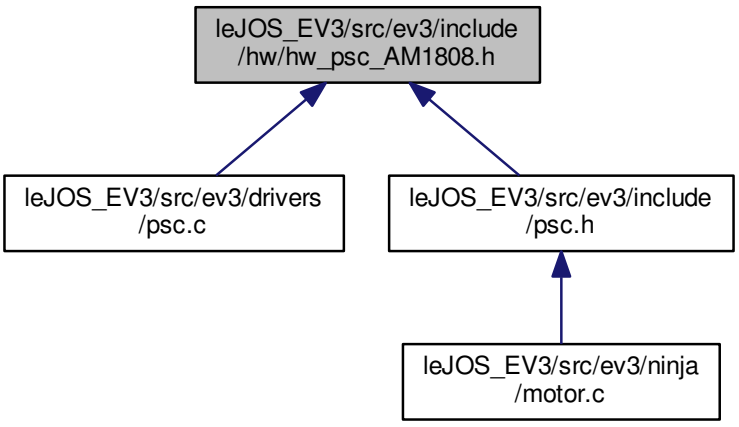
### 4.36.1 Detailed Description

GPIO register definitions.

## 4.37 leJOS_EV3/src/ev3/include/hw/hw_psc_AM1808.h File Reference

PSC register definitions for AM1808.

This graph shows which files directly or indirectly include this file:



## Macros

- #define **PSC_POWERDOMAIN_ALWAYS_ON** 0
- #define **PSC_REVID** (0x0)
- #define **PSC_INTEVAL** (0x18)
- #define **PSC_MERRPR0** (0x40)
- #define **PSC_MERRCR0** (0x50)
- #define **PSC_PERRPR** (0x60)
- #define **PSC_PERRCR** (0x68)
- #define **PSC_PTCMD** (0x120)
- #define **PSC_PTSTAT** (0x128)
- #define **PSC_PDSTAT0** (0x200)
- #define **PSC_PDSTAT1** (0x204)
- #define **PSC_PDCTL0** (0x300)
- #define **PSC_PDCTL1** (0x304)
- #define **PSC_PDCFG0** (0x400)
- #define **PSC_PDCFG1** (0x404)
- #define **PSC_MDSTAT**(n) (0x800 + (n ∗ 4))
- #define **PSC_MDCTL**(n) (0xA00 + (n ∗ 4))
- #define **PSC_REVID_REV** (0xFFFFFFFFu)
- #define **PSC_REVID_REV_SHIFT** (0x00000000u)
- #define **PSC_INTEVAL_ALLEV** (0x00000001u)
- #define **PSC_INTEVAL_ALLEV_SHIFT** (0x00000000u)
- #define **PSC_MERRPR0_M15** (0x0000C000u)
- #define **PSC_MERRPR0_M15_SHIFT** (0x0000000Eu)
- #define **PSC_MERRPR0_M14** (0x00006000u)
- #define **PSC_MERRPR0_M14_SHIFT** (0x0000000Du)
- #define **PSC_MERRCR0_M15** (0x0000C000u)
- #define **PSC_MERRCR0_M15_SHIFT** (0x0000000Eu)
- #define **PSC_MERRCR0_M14** (0x00006000u)
- #define **PSC_MERRCR0_M14_SHIFT** (0x0000000Du)
- #define **PSC_PERRPR_P1** (0x00000002u)
- #define **PSC_PERRPR_P1_SHIFT** (0x00000001u)
- #define **PSC_PERRPR_P0** (0x00000001u)
- #define **PSC_PERRPR_P0_SHIFT** (0x00000000u)
- #define **PSC_PERRCR_P1** (0x00000002u)
- #define **PSC_PERRCR_P1_SHIFT** (0x00000001u)
- #define **PSC_PERRCR_P0** (0x00000001u)
- #define **PSC_PERRCR_P0_SHIFT** (0x00000000u)
- #define **PSC_PTCMD_GO1** (0x00000002u)
- #define **PSC_PTCMD_GO1_SHIFT** (0x00000001u)
- #define **PSC_PTCMD_GO0** (0x00000001u)
- #define **PSC_PTCMD_GO0_SHIFT** (0x00000000u)
- #define **PSC_PTSTAT_GOSTAT1** (0x00000002u)
- #define **PSC_PTSTAT_GOSTAT1_SHIFT** (0x00000001u)
- #define **PSC_PTSTAT_GOSTAT0** (0x00000001u)
- #define **PSC_PTSTAT_GOSTAT0_SHIFT** (0x00000000u)
- #define **PSC_PDSTAT0_EMUIHB** (0x00000800u)
- #define **PSC_PDSTAT0_EMUIHB_SHIFT** (0x0000000Bu)
- #define **PSC_PDSTAT0_STATE** (0x0000001Fu)
- #define **PSC_PDSTAT0_STATE_SHIFT** (0x00000000u)
- #define **PSC_PDSTAT1_EMUIHB** (0x00000800u)
- #define **PSC_PDSTAT1_EMUIHB_SHIFT** (0x0000000Bu)
- #define **PSC_PDSTAT1_STATE** (0x0000001Fu)
- #define **PSC_PDSTAT1_STATE_SHIFT** (0x00000000u)

- #define **PSC_PDCTL0_WAKECNT** (0x00FF0000u)
- #define **PSC_PDCTL0_WAKECNT_SHIFT** (0x00000010u)
- #define **PSC_PDCTL0_PDMODE** (0x0000F000u)
- #define **PSC_PDCTL0_PDMODE_SHIFT** (0x0000000Cu)
- #define **PSC_PDCTL0_EMUIHBIE** (0x00000200u)
- #define **PSC_PDCTL0_EMUIHBIE_SHIFT** (0x00000009u)
- #define **PSC_PDCTL0_NEXT** (0x00000001u)
- #define **PSC_PDCTL0_NEXT_SHIFT** (0x00000000u)
- #define **PSC_PDCTL1_WAKECNT** (0x00FF0000u)
- #define **PSC_PDCTL1_WAKECNT_SHIFT** (0x00000010u)
- #define **PSC_PDCTL1_PDMODE** (0x0000F000u)
- #define **PSC_PDCTL1_PDMODE_SHIFT** (0x0000000Cu)
- #define **PSC_PDCTL1_PDMODE_OFF** (0x00000000u)
- #define **PSC_PDCTL1_PDMODE_RAM_OFF** (0x00000008u)
- #define **PSC_PDCTL1_PDMODE_DEEP_SLEEP** (0x00000009u)
- #define **PSC_PDCTL1_PDMODE_LIGHT_SLEEP** (0x0000000Au)
- #define **PSC_PDCTL1_PDMODE_RETENTION** (0x0000000Bu)
- #define **PSC_PDCTL1_PDMODE_ON** (0x0000000Fu)
- #define **PSC_PDCTL1_EMUIHBIE** (0x00000200u)
- #define **PSC_PDCTL1_EMUIHBIE_SHIFT** (0x00000009u)
- #define **PSC_PDCTL1_NEXT** (0x00000001u)
- #define **PSC_PDCTL1_NEXT_SHIFT** (0x00000000u)
- #define **PSC_PDCFG0_PDLOCK** (0x00000008u)
- #define **PSC_PDCFG0_PDLOCK_SHIFT** (0x00000003u)
- #define **PSC_PDCFG0_ICEPICK** (0x00000004u)
- #define **PSC_PDCFG0_ICEPICK_SHIFT** (0x00000002u)
- #define **PSC_PDCFG0_RAM_PSM** (0x00000002u)
- #define **PSC_PDCFG0_RAM_PSM_SHIFT** (0x00000001u)
- #define **PSC_PDCFG0_ALWAYSON** (0x00000001u)
- #define **PSC_PDCFG0_ALWAYSON_SHIFT** (0x00000000u)
- #define **PSC_PDCFG1_PDLOCK** (0x00000008u)
- #define **PSC_PDCFG1_PDLOCK_SHIFT** (0x00000003u)
- #define **PSC_PDCFG1_ICEPICK** (0x00000004u)
- #define **PSC_PDCFG1_ICEPICK_SHIFT** (0x00000002u)
- #define **PSC_PDCFG1_RAM_PSM** (0x00000002u)
- #define **PSC_PDCFG1_RAM_PSM_SHIFT** (0x00000001u)
- #define **PSC_PDCFG1_ALWAYSON** (0x00000001u)
- #define **PSC_PDCFG1_ALWAYSON_SHIFT** (0x00000000u)
- #define **PSC_MDSTAT_EMUIHB** (0x00020000u)
- #define **PSC_MDSTAT_EMUIHB_SHIFT** (0x00000011u)
- #define **PSC_MDSTAT_EMURST** (0x00010000u)
- #define **PSC_MDSTAT_EMURST_SHIFT** (0x00000010u)
- #define **PSC_MDSTAT_MCKOUT** (0x00001000u)
- #define **PSC_MDSTAT_MCKOUT_SHIFT** (0x0000000Cu)
- #define **PSC_MDSTAT_MRSTDONE** (0x00000800u)
- #define **PSC_MDSTAT_MRSTDONE_SHIFT** (0x0000000Bu)
- #define **PSC_MDSTAT_MRST** (0x00000400u)
- #define **PSC_MDSTAT_MRST_SHIFT** (0x0000000Au)
- #define **PSC_MDSTAT_LRSTDONE** (0x00000200u)
- #define **PSC_MDSTAT_LRSTDONE_SHIFT** (0x00000009u)
- #define **PSC_MDSTAT_LRST** (0x00000100u)
- #define **PSC_MDSTAT_LRST_SHIFT** (0x00000008u)
- #define **PSC_MDSTAT_STATE** (0x0000003Fu)
- #define **PSC_MDSTAT_STATE_SHIFT** (0x00000000u)
- #define **PSC_MDSTAT_STATE_SWRSTDISABLE** (0x00000000u)

- #define **PSC_MDSTAT_STATE_SYNCRST** (0x00000001u)
- #define **PSC_MDSTAT_STATE_AUTOSLEEP** (0x00000004u)
- #define **PSC_MDSTAT_STATE_AUTOWAKE** (0x00000005u)
- #define **PSC_MDCTL_FORCE** (0x80000000u)
- #define **PSC_MDCTL_FORCE_SHIFT** (0x0000001Fu)
- #define **PSC_MDCTL_EMUIHBIE** (0x00000400u)
- #define **PSC_MDCTL_EMUIHBIE_SHIFT** (0x0000000Au)
- #define **PSC_MDCTL_EMURSTIE** (0x00000200u)
- #define **PSC_MDCTL_EMURSTIE_SHIFT** (0x00000009u)
- #define **PSC_MDCTL_LRST** (0x00000100u)
- #define **PSC_MDCTL_LRST_SHIFT** (0x00000008u)
- #define **PSC_MDCTL_NEXT** (0x0000001Fu)
- #define **PSC_MDCTL_NEXT_SHIFT** (0x00000000u)
- #define **PSC_MDCTL_NEXT_SWRSTDISABLE** (0x00000000u)
- #define **PSC_MDCTL_NEXT_SYNCRST** (0x00000001u)
- #define **PSC_MDCTL_NEXT_DISABLE** (0x00000002u)
- #define **PSC_MDCTL_NEXT_ENABLE** (0x00000003u)
- #define **PSC_MDCTL_NEXT_AUTOWAKE** (0x00000005u)

### Enumerations

- enum **Psc0Peripheral** {
  **HW_PSC_CC0** = 0, **HW_PSC_TC0** = 1, **HW_PSC_TC1** = 2, **HW_PSC_EMIFA** = 3,
  **HW_PSC_SPI0** = 4, **HW_PSC_MMCSD0** = 5, **HW_PSC_AINTC** = 6, **HW_PSC_ARM_RAMROM** = 7,
  **HW_PSC_UART0** = 9, **HW_PSC_SCR0_SS** = 10, **HW_PSC_SCR1_SS** = 11, **HW_PSC_SCR2_SS** = 12,
  **HW_PSC_PRU** = 13, **HW_PSC_ARM** = 14, **HW_PSC_DSP** = 15 }
- enum **Psc1Peripheral** {
  **HW_PSC_CC1** = 0, **HW_PSC_USB0** = 1, **HW_PSC_USB1** = 2, **HW_PSC_GPIO** = 3,
  **HW_PSC_UHPI** = 4, **HW_PSC_EMAC** = 5, **HW_PSC_DDR2_MDDR** = 6, **HW_PSC_MCASP0** = 7,
  **HW_PSC_SATA** = 8, **HW_PSC_VPIF** = 9, **HW_PSC_SPI1** = 10, **HW_PSC_I2C1** = 11,
  **HW_PSC_UART1** = 12, **HW_PSC_UART2** = 13, **HW_PSC_MCBSP0** = 14, **HW_PSC_MCBSP1** = 15,
  **HW_PSC_LCDC** = 16, **HW_PSC_EHRPWM** = 17, **HW_PSC_MMCSD1** = 18, **HW_PSC_UPP** = 19,
  **HW_PSC_ECAP0_1_2** = 20, **HW_PSC_TC2** = 21, **HW_PSC_SCRF0_SS** = 24, **HW_PSC_SCRF1_SS** = 25,
  **HW_PSC_SCRF2_SS** = 26, **HW_PSC_SCRF6_SS** = 27, **HW_PSC_SCRF7_SS** = 28, **HW_PSC_SCRF8_SS** = 29,
  **HW_PSC_BR_F7** = 30, **HW_PSC_SHRAM** = 31 }

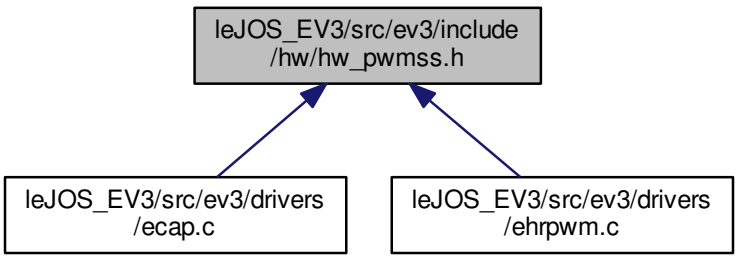
#### 4.37.1 Detailed Description

PSC register definitions for AM1808.

## 4.38 leJOS_EV3/src/ev3/include/hw/hw_pwmss.h File Reference

PWMSS register definitions.

This graph shows which files directly or indirectly include this file:




### Macros

- #define **PWMSS_CLOCK_CONFIG** 0x08
- #define **PWMSS_CLOCK_STATUS** 0x0C
- #define **PWMSS_ECAP_CLK_EN_ACK_SHIFT** 0x00
- #define **PWMSS_ECAP_CLK_STOP_ACK_SHIFT** 0x01
- #define **PWMSS_EHRPWM_CLK_EN_ACK_SHIFT** 0x08
- #define **PWMSS_EHRPWM_CLK_STOP_ACK_SHIFT** 0x09
- #define **PWMSS_ECAP_CLK_EN_ACK** 0x01
- #define **PWMSS_ECAP_CLK_STOP_ACK** 0x02
- #define **PWMSS_EHRPWM_CLK_EN_ACK** 0x100
- #define **PWMSS_EHRPWM_CLK_STOP_ACK** 0x200

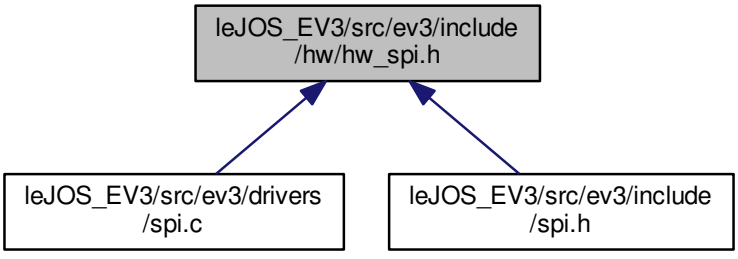
#### 4.38.1 Detailed Description

PWMSS register definitions.

## 4.39 leJOS_EV3/src/ev3/include/hw/hw_spi.h File Reference

SPI register definitions.

This graph shows which files directly or indirectly include this file:



## Macros

- #define **SPI_SPIGCR0** (0x0)
- #define **SPI_SPIGCR1** (0x4)
- #define **SPI_SPIINT0** (0x8)
- #define **SPI_SPILVL** (0xC)
- #define **SPI_SPIFLG** (0x10)
- #define **SPI_SPIPC**(n) (0x14 + (4 ∗ n))
- #define **SPI_SPIDAT0** (0x38)
- #define **SPI_SPIDAT1** (0x3C)
- #define **SPI_SPIBUF** (0x40)
- #define **SPI_SPIEMU** (0x44)
- #define **SPI_SPIDELAY** (0x48)
- #define **SPI_SPIDEF** (0x4C)
- #define **SPI_SPIFMT**(n) (0x50 + (n ∗ 4))
- #define **SPI_INTVEC1** (0x64)
- #define **SPI_SPIGCR0_RESET** (0x00000001u)
- #define **SPI_SPIGCR0_RESET_SHIFT** (0x00000000u)
- #define **SPI_SPIGCR1_ENABLE** (0x01000000)
- #define **SPI_SPIGCR1_ENABLE_SHIFT** (0x00000018u)
- #define **SPI_SPIGCR1_LOOPBACK** (0x00010000u)
- #define **SPI_SPIGCR1_LOOPBACK_SHIFT** (0x00000010u)
- #define **SPI_SPIGCR1_POWERDOWN** (0x00000100u)
- #define **SPI_SPIGCR1_POWERDOWN_SHIFT** (0x00000008u)
- #define **SPI_SPIGCR1_CLKMOD** (0x00000002u)
- #define **SPI_SPIGCR1_CLKMOD_SHIFT** (0x00000001u)
- #define **SPI_SPIGCR1_MASTER** (0x00000001u)
- #define **SPI_SPIGCR1_MASTER_SHIFT** (0x00000000u)
- #define **SPI_SPIINT0_ENABLEHIGHZ** (0x01000000u)
- #define **SPI_SPIINT0_ENABLEHIGHZ_SHIFT** (0x00000018u)
- #define **SPI_SPIINT0_DMAREQEN** (0x00010000u)
- #define **SPI_SPIINT0_DMAREQEN_SHIFT** (0x00000010u)
- #define **SPI_SPIINT0_TXINTENA** (0x00000200u)
- #define **SPI_SPIINT0_TXINTENA_SHIFT** (0x00000009u)
- #define **SPI_SPIINT0_RXINTENA** (0x00000100u)
- #define **SPI_SPIINT0_RXINTENA_SHIFT** (0x00000008u)
- #define **SPI_SPIINT0_OVRNINTENA** (0x00000040u)
- #define **SPI_SPIINT0_OVRNINTENA_SHIFT** (0x00000006u)
- #define **SPI_SPIINT0_BITERRENA** (0x00000010u)
- #define **SPI_SPIINT0_BITERRENA_SHIFT** (0x00000004u)
- #define **SPI_SPIINT0_DESYNCENA** (0x00000008u)
- #define **SPI_SPIINT0_DESYNCENA_SHIFT** (0x00000003u)
- #define **SPI_SPIINT0_PARERRENA** (0x00000004u)
- #define **SPI_SPIINT0_PARERRENA_SHIFT** (0x00000002u)
- #define **SPI_SPIINT0_TIMEOUTENA** (0x00000002u)
- #define **SPI_SPIINT0_TIMEOUTENA_SHIFT** (0x00000001u)
- #define **SPI_SPIINT0_DLENERRENA** (0x00000001u)
- #define **SPI_SPIINT0_DLENERRENA_SHIFT** (0x00000000u)
- #define **SPI_SPILVL_TXINTLVL** (0x00000200u)
- #define **SPI_SPILVL_TXINTLVL_SHIFT** (0x00000009u)
- #define **SPI_SPILVL_RXINTLVL** (0x00000100u)
- #define **SPI_SPILVL_RXINTLVL_SHIFT** (0x00000008u)
- #define **SPI_SPILVL_OVRNINTLVL** (0x00000040u)
- #define **SPI_SPILVL_OVRNINTLVL_SHIFT** (0x00000006u)
- #define **SPI_SPILVL_BITERRLVL** (0x00000010u)

- #define **SPI_SPILVL_BITERRLVL_SHIFT** (0x00000004u)
- #define **SPI_SPILVL_DESYNCLVL** (0x00000008u)
- #define **SPI_SPILVL_DESYNCLVL_SHIFT** (0x00000003u)
- #define **SPI_SPILVL_PARERRLVL** (0x00000004u)
- #define **SPI_SPILVL_PARERRLVL_SHIFT** (0x00000002u)
- #define **SPI_SPILVL_TIMEOUTLVL** (0x00000002u)
- #define **SPI_SPILVL_TIMEOUTLVL_SHIFT** (0x00000001u)
- #define **SPI_SPILVL_DLENERRLVL** (0x00000001u)
- #define **SPI_SPILVL_DLENERRLVL_SHIFT** (0x00000000u)
- #define **SPI_SPIFLG_TXINTFLG** (0x00000200u)
- #define **SPI_SPIFLG_TXINTFLG_SHIFT** (0x00000009u)
- #define **SPI_SPIFLG_RXINTFLG** (0x00000100u)
- #define **SPI_SPIFLG_RXINTFLG_SHIFT** (0x00000008u)
- #define **SPI_SPIFLG_OVRNINTFLG** (0x00000040u)
- #define **SPI_SPIFLG_OVRNINTFLG_SHIFT** (0x00000006u)
- #define **SPI_SPIFLG_BITERRFLG** (0x00000010u)
- #define **SPI_SPIFLG_BITERRFLG_SHIFT** (0x00000004u)
- #define **SPI_SPIFLG_DESYNCFLG** (0x00000008u)
- #define **SPI_SPIFLG_DESYNCFLG_SHIFT** (0x00000003u)
- #define **SPI_SPIFLG_PARERRFLG** (0x00000004u)
- #define **SPI_SPIFLG_PARERRFLG_SHIFT** (0x00000002u)
- #define **SPI_SPIFLG_TIMEOUTFLG** (0x00000002u)
- #define **SPI_SPIFLG_TIMEOUTFLG_SHIFT** (0x00000001u)
- #define **SPI_SPIFLG_DLENERRFLG** (0x00000001u)
- #define **SPI_SPIFLG_DLENERRFLG_SHIFT** (0x00000000u)
- #define **SPI_SPIPC0_SOMIFUN** (0x00000800u)
- #define **SPI_SPIPC0_SOMIFUN_SHIFT** (0x0000000Bu)
- #define **SPI_SPIPC0_SIMOFUN** (0x00000400u)
- #define **SPI_SPIPC0_SIMOFUN_SHIFT** (0x0000000Au)
- #define **SPI_SPIPC0_CLKFUN** (0x00000200u)
- #define **SPI_SPIPC0_CLKFUN_SHIFT** (0x00000009u)
- #define **SPI_SPIPC0_ENAFUN** (0x00000100u)
- #define **SPI_SPIPC0_ENAFUN_SHIFT** (0x00000008u)
- #define **SPI_SPIPC0_SCS0FUN7** (0x00000080u)
- #define **SPI_SPIPC0_SCS0FUN7_SHIFT** (0x00000007u)
- #define **SPI_SPIPC0_SCS0FUN6** (0x00000040u)
- #define **SPI_SPIPC0_SCS0FUN6_SHIFT** (0x00000006u)
- #define **SPI_SPIPC0_SCS0FUN5** (0x00000020u)
- #define **SPI_SPIPC0_SCS0FUN5_SHIFT** (0x00000005u)
- #define **SPI_SPIPC0_SCS0FUN4** (0x00000010u)
- #define **SPI_SPIPC0_SCS0FUN4_SHIFT** (0x00000004u)
- #define **SPI_SPIPC0_SCS0FUN3** (0x00000008u)
- #define **SPI_SPIPC0_SCS0FUN3_SHIFT** (0x00000003u)
- #define **SPI_SPIPC0_SCS0FUN2** (0x00000004u)
- #define **SPI_SPIPC0_SCS0FUN2_SHIFT** (0x00000002u)
- #define **SPI_SPIPC0_SCS0FUN1** (0x00000002u)
- #define **SPI_SPIPC0_SCS0FUN1_SHIFT** (0x00000001u)
- #define **SPI_SPIPC0_SCS0FUN0** (0x00000001u)
- #define **SPI_SPIPC0_SCS0FUN0_SHIFT** (0x00000000u)
- #define **SPI_SPIPC1_SOMIDIR** (0x00000800u)
- #define **SPI_SPIPC1_SOMIDIR_SHIFT** (0x0000000Bu)
- #define **SPI_SPIPC1_SIMODIR** (0x00000400u)
- #define **SPI_SPIPC1_SIMODIR_SHIFT** (0x0000000Au)
- #define **SPI_SPIPC1_CLKDIR** (0x00000200u)
- #define **SPI_SPIPC1_CLKDIR_SHIFT** (0x00000009u)

- #define **SPI_SPIPC1_ENADIR** (0x00000100u)
- #define **SPI_SPIPC1_ENADIR_SHIFT** (0x00000008u)
- #define **SPI_SPIPC1_SCS0DIR7** (0x00000080u)
- #define **SPI_SPIPC1_SCS0DIR7_SHIFT** (0x00000007u)
- #define **SPI_SPIPC1_SCS0DIR6** (0x00000040u)
- #define **SPI_SPIPC1_SCS0DIR6_SHIFT** (0x00000006u)
- #define **SPI_SPIPC1_SCS0DIR5** (0x00000020u)
- #define **SPI_SPIPC1_SCS0DIR5_SHIFT** (0x00000005u)
- #define **SPI_SPIPC1_SCS0DIR4** (0x00000010u)
- #define **SPI_SPIPC1_SCS0DIR4_SHIFT** (0x00000004u)
- #define **SPI_SPIPC1_SCS0DIR3** (0x00000008u)
- #define **SPI_SPIPC1_SCS0DIR3_SHIFT** (0x00000003u)
- #define **SPI_SPIPC1_SCS0DIR2** (0x00000004u)
- #define **SPI_SPIPC1_SCS0DIR2_SHIFT** (0x00000002u)
- #define **SPI_SPIPC1_SCS0DIR1** (0x00000002u)
- #define **SPI_SPIPC1_SCS0DIR1_SHIFT** (0x00000001u)
- #define **SPI_SPIPC1_SCS0DIR0** (0x00000001u)
- #define **SPI_SPIPC1_SCS0DIR0_SHIFT** (0x00000000u)
- #define **SPI_SPIPC2_SOMIDIN** (0x00000800u)
- #define **SPI_SPIPC2_SOMIDIN_SHIFT** (0x0000000Bu)
- #define **SPI_SPIPC2_SIMODIN** (0x00000400u)
- #define **SPI_SPIPC2_SIMODIN_SHIFT** (0x0000000Au)
- #define **SPI_SPIPC2_CLKDIN** (0x00000200u)
- #define **SPI_SPIPC2_CLKDIN_SHIFT** (0x00000009u)
- #define **SPI_SPIPC2_ENADIN** (0x00000100u)
- #define **SPI_SPIPC2_ENADIN_SHIFT** (0x00000008u)
- #define **SPI_SPIPC2_SCS0DIN7** (0x00000080u)
- #define **SPI_SPIPC2_SCS0DIN7_SHIFT** (0x00000007u)
- #define **SPI_SPIPC2_SCS0DIN6** (0x00000040u)
- #define **SPI_SPIPC2_SCS0DIN6_SHIFT** (0x00000006u)
- #define **SPI_SPIPC2_SCS0DIN5** (0x00000020u)
- #define **SPI_SPIPC2_SCS0DIN5_SHIFT** (0x00000005u)
- #define **SPI_SPIPC2_SCS0DIN4** (0x00000010u)
- #define **SPI_SPIPC2_SCS0DIN4_SHIFT** (0x00000004u)
- #define **SPI_SPIPC2_SCS0DIN3** (0x00000008u)
- #define **SPI_SPIPC2_SCS0DIN3_SHIFT** (0x00000003u)
- #define **SPI_SPIPC2_SCS0DIN2** (0x00000004u)
- #define **SPI_SPIPC2_SCS0DIN2_SHIFT** (0x00000002u)
- #define **SPI_SPIPC2_SCS0DIN1** (0x00000002u)
- #define **SPI_SPIPC2_SCS0DIN1_SHIFT** (0x00000001u)
- #define **SPI_SPIPC2_SCS0DIN0** (0x00000001u)
- #define **SPI_SPIPC2_SCS0DIN0_SHIFT** (0x00000000u)
- #define **SPI_SPIPC3_SOMIDOUT** (0x00000800u)
- #define **SPI_SPIPC3_SOMIDOUT_SHIFT** (0x0000000Bu)
- #define **SPI_SPIPC3_SIMODOUT** (0x00000400u)
- #define **SPI_SPIPC3_SIMODOUT_SHIFT** (0x0000000Au)
- #define **SPI_SPIPC3_CLKDOUT** (0x00000200u)
- #define **SPI_SPIPC3_CLKDOUT_SHIFT** (0x00000009u)
- #define **SPI_SPIPC3_ENADOUT** (0x00000100u)
- #define **SPI_SPIPC3_ENADOUT_SHIFT** (0x00000008u)
- #define **SPI_SPIPC3_SCS0DOUT7** (0x00000080u)
- #define **SPI_SPIPC3_SCS0DOUT7_SHIFT** (0x00000007u)
- #define **SPI_SPIPC3_SCS0DOUT6** (0x00000040u)
- #define **SPI_SPIPC3_SCS0DOUT6_SHIFT** (0x00000006u)
- #define **SPI_SPIPC3_SCS0DOUT5** (0x00000020u)

- #define **SPI_SPIPC3_SCS0DOUT5_SHIFT** (0x00000005u)
- #define **SPI_SPIPC3_SCS0DOUT4** (0x00000010u)
- #define **SPI_SPIPC3_SCS0DOUT4_SHIFT** (0x00000004u)
- #define **SPI_SPIPC3_SCS0DOUT3** (0x00000008u)
- #define **SPI_SPIPC3_SCS0DOUT3_SHIFT** (0x00000003u)
- #define **SPI_SPIPC3_SCS0DOUT2** (0x00000004u)
- #define **SPI_SPIPC3_SCS0DOUT2_SHIFT** (0x00000002u)
- #define **SPI_SPIPC3_SCS0DOUT1** (0x00000002u)
- #define **SPI_SPIPC3_SCS0DOUT1_SHIFT** (0x00000001u)
- #define **SPI_SPIPC3_SCS0DOUT0** (0x00000001u)
- #define **SPI_SPIPC3_SCS0DOUT0_SHIFT** (0x00000000u)
- #define **SPI_SPIPC4_SOMISET** (0x00000800u)
- #define **SPI_SPIPC4_SOMISET_SHIFT** (0x0000000Bu)
- #define **SPI_SPIPC4_SIMOSET** (0x00000400u)
- #define **SPI_SPIPC4_SIMOSET_SHIFT** (0x0000000Au)
- #define **SPI_SPIPC4_CLKSET** (0x00000200u)
- #define **SPI_SPIPC4_CLKSET_SHIFT** (0x00000009u)
- #define **SPI_SPIPC4_ENASET** (0x00000100u)
- #define **SPI_SPIPC4_ENASET_SHIFT** (0x00000008u)
- #define **SPI_SPIPC4_SCS0SET7** (0x00000080u)
- #define **SPI_SPIPC4_SCS0SET7_SHIFT** (0x00000007u)
- #define **SPI_SPIPC4_SCS0SET6** (0x00000040u)
- #define **SPI_SPIPC4_SCS0SET6_SHIFT** (0x00000006u)
- #define **SPI_SPIPC4_SCS0SET5** (0x00000020u)
- #define **SPI_SPIPC4_SCS0SET5_SHIFT** (0x00000005u)
- #define **SPI_SPIPC4_SCS0SET4** (0x00000010u)
- #define **SPI_SPIPC4_SCS0SET4_SHIFT** (0x00000004u)
- #define **SPI_SPIPC4_SCS0SET3** (0x00000008u)
- #define **SPI_SPIPC4_SCS0SET3_SHIFT** (0x00000003u)
- #define **SPI_SPIPC4_SCS0SET2** (0x00000004u)
- #define **SPI_SPIPC4_SCS0SET2_SHIFT** (0x00000002u)
- #define **SPI_SPIPC4_SCS0SET1** (0x00000002u)
- #define **SPI_SPIPC4_SCS0SET1_SHIFT** (0x00000001u)
- #define **SPI_SPIPC4_SCS0SET0** (0x00000001u)
- #define **SPI_SPIPC4_SCS0SET0_SHIFT** (0x00000000u)
- #define **SPI_SPIPC5_SOMICLR** (0x00000800u)
- #define **SPI_SPIPC5_SOMICLR_SHIFT** (0x0000000Bu)
- #define **SPI_SPIPC5_SIMOCLR** (0x00000400u)
- #define **SPI_SPIPC5_SIMOCLR_SHIFT** (0x0000000Au)
- #define **SPI_SPIPC5_CLKCLR** (0x00000200u)
- #define **SPI_SPIPC5_CLKCLR_SHIFT** (0x00000009u)
- #define **SPI_SPIPC5_ENACLR** (0x00000100u)
- #define **SPI_SPIPC5_ENACLR_SHIFT** (0x00000008u)
- #define **SPI_SPIPC5_SCS0CLR7** (0x00000080u)
- #define **SPI_SPIPC5_SCS0CLR7_SHIFT** (0x00000007u)
- #define **SPI_SPIPC5_SCS0CLR6** (0x00000040u)
- #define **SPI_SPIPC5_SCS0CLR6_SHIFT** (0x00000006u)
- #define **SPI_SPIPC5_SCS0CLR5** (0x00000020u)
- #define **SPI_SPIPC5_SCS0CLR5_SHIFT** (0x00000005u)
- #define **SPI_SPIPC5_SCS0CLR4** (0x00000010u)
- #define **SPI_SPIPC5_SCS0CLR4_SHIFT** (0x00000004u)
- #define **SPI_SPIPC5_SCS0CLR3** (0x00000008u)
- #define **SPI_SPIPC5_SCS0CLR3_SHIFT** (0x00000003u)
- #define **SPI_SPIPC5_SCS0CLR2** (0x00000004u)
- #define **SPI_SPIPC5_SCS0CLR2_SHIFT** (0x00000002u)

- #define **SPI_SPIPC5_SCS0CLR1** (0x00000002u)
- #define **SPI_SPIPC5_SCS0CLR1_SHIFT** (0x00000001u)
- #define **SPI_SPIPC5_SCS0CLR0** (0x00000001u)
- #define **SPI_SPIPC5_SCS0CLR0_SHIFT** (0x00000000u)
- #define **SPI_SPIDAT0_TXDATA** (0x0000FFFFu)
- #define **SPI_SPIDAT0_TXDATA_SHIFT** (0x00000000u)
- #define **SPI_SPIDAT1_CSHOLD** (0x10000000u)
- #define **SPI_SPIDAT1_CSHOLD_SHIFT** (0x0000001Cu)
- #define **SPI_SPIDAT1_WDEL** (0x04000000u)
- #define **SPI_SPIDAT1_WDEL_SHIFT** (0x0000001Au)
- #define **SPI_SPIDAT1_DFSEL** (0x03000000u)
- #define **SPI_SPIDAT1_DFSEL_SHIFT** (0x00000018u)
- #define **SPI_SPIDAT1_DFSEL_FORMAT0** (0x00000000u)
- #define **SPI_SPIDAT1_DFSEL_FORMAT1** (0x00000001u)
- #define **SPI_SPIDAT1_DFSEL_FORMAT2** (0x00000002u)
- #define **SPI_SPIDAT1_DFSEL_FORMAT3** (0x00000003u)
- #define **SPI_SPIDAT1_CSNR** (0x00FF0000u)
- #define **SPI_SPIDAT1_CSNR_SHIFT** (0x00000010u)
- #define **SPI_SPIDAT1_TXDATA** (0x0000FFFFu)
- #define **SPI_SPIDAT1_TXDATA_SHIFT** (0x00000000u)
- #define **SPI_SPIBUF_RXEMPTY** (0x80000000u)
- #define **SPI_SPIBUF_RXEMPTY_SHIFT** (0x0000001Fu)
- #define **SPI_SPIBUF_RXOVR** (0x40000000u)
- #define **SPI_SPIBUF_RXOVR_SHIFT** (0x0000001Eu)
- #define **SPI_SPIBUF_TXFULL** (0x20000000u)
- #define **SPI_SPIBUF_TXFULL_SHIFT** (0x0000001Du)
- #define **SPI_SPIBUF_BITERR** (0x10000000u)
- #define **SPI_SPIBUF_BITERR_SHIFT** (0x0000001Cu)
- #define **SPI_SPIBUF_DESYNC** (0x08000000u)
- #define **SPI_SPIBUF_DESYNC_SHIFT** (0x0000001Bu)
- #define **SPI_SPIBUF_PARERR** (0x04000000u)
- #define **SPI_SPIBUF_PARERR_SHIFT** (0x0000001Au)
- #define **SPI_SPIBUF_TIMEOUT** (0x02000000u)
- #define **SPI_SPIBUF_TIMEOUT_SHIFT** (0x00000019u)
- #define **SPI_SPIBUF_DLENERR** (0x01000000u)
- #define **SPI_SPIBUF_DLENERR_SHIFT** (0x00000018u)
- #define **SPI_SPIBUF_RXDATA** (0x0000FFFFu)
- #define **SPI_SPIBUF_RXDATA_SHIFT** (0x00000000u)
- #define **SPI_SPIEMU_RXDATA** (0x0000FFFFu)
- #define **SPI_SPIEMU_RXDATA_SHIFT** (0x00000000u)
- #define **SPI_SPIDELAY_C2TDELAY** (0xFF000000u)
- #define **SPI_SPIDELAY_C2TDELAY_SHIFT** (0x00000018u)
- #define **SPI_SPIDELAY_T2CDELAY** (0x00FF0000u)
- #define **SPI_SPIDELAY_T2CDELAY_SHIFT** (0x00000010u)
- #define **SPI_SPIDELAY_T2EDELAY** (0x0000FF00u)
- #define **SPI_SPIDELAY_T2EDELAY_SHIFT** (0x00000008u)
- #define **SPI_SPIDELAY_C2EDELAY** (0x000000FFu)
- #define **SPI_SPIDELAY_C2EDELAY_SHIFT** (0x00000000u)
- #define **SPI_SPIDEF_CSDEF** (0x0000000FFu)
- #define **SPI_SPIDEF_CSDEFN**(n) (1 << n)
- #define **SPI_SPIDEF_CSDEF_SHIFT** (0x00000000u)
- #define **SPI_SPIFMT_WDELAY** (0x3F000000u)
- #define **SPI_SPIFMT_WDELAY_SHIFT** (0x00000018u)
- #define **SPI_SPIFMT_PARPOL** (0x00800000u)
- #define **SPI_SPIFMT_PARPOL_SHIFT** (0x00000017u)

- #define **SPI_SPIFMT_PARENA** (0x00400000u)
- #define **SPI_SPIFMT_PARENA_SHIFT** (0x00000016u)
- #define **SPI_SPIFMT_WAITENA** (0x00200000u)
- #define **SPI_SPIFMT_WAITENA_SHIFT** (0x00000015u)
- #define **SPI_SPIFMT_SHIFTDIR** (0x00100000u)
- #define **SPI_SPIFMT_SHIFTDIR_SHIFT** (0x00000014u)
- #define **SPI_SPIFMT_DISCSTIMERS** (0x00040000u)
- #define **SPI_SPIFMT_DISCSTIMERS_SHIFT** (0x00000012u)
- #define **SPI_SPIFMT_POLARITY** (0x00020000u)
- #define **SPI_SPIFMT_POLARITY_SHIFT** (0x00000011u)
- #define **SPI_SPIFMT_PHASE** (0x00010000u)
- #define **SPI_SPIFMT_PHASE_SHIFT** (0x00000010u)
- #define **SPI_SPIFMT_PRESCALE** (0x0000FF00u)
- #define **SPI_SPIFMT_PRESCALE_SHIFT** (0x00000008u)
- #define **SPI_SPIFMT_CHARLEN** (0x0000001Fu)
- #define **SPI_SPIFMT_CHARLEN_SHIFT** (0x00000000u)
- #define **SPI_INTVEC_INTVECT** (0x0000003Eu)
- #define **SPI_INTVEC_INTVECT_SHIFT** (0x00000001u)

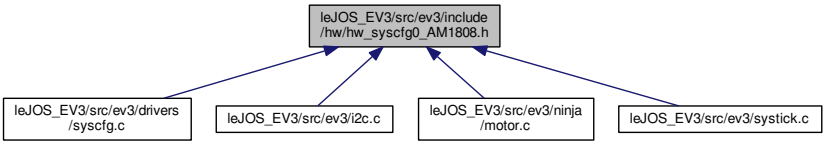
### 4.39.1 Detailed Description

SPI register definitions.

## 4.40 leJOS_EV3/src/ev3/include/hw/hw_syscfg0_AM1808.h File Reference

SYSCFG0 register definitions for AM1808.

This graph shows which files directly or indirectly include this file:




### Macros

- #define **SYSCFG0_REVID** (0x0)
- #define **SYSCFG0_DIEIDR0** (0x8)
- #define **SYSCFG0_DIEIDR1** (0xC)
- #define **SYSCFG0_DIEIDR2** (0x10)
- #define **SYSCFG0_DIEIDR3** (0x14)
- #define **SYSCFG0_DEVIDR0** (0x18)
- #define **SYSCFG0_BOOTCFG** (0x20)
- #define **SYSCFG0_KICK0R** (0x38)
- #define **SYSCFG0_KICK1R** (0x3C)
- #define **SYSCFG0_HOST0CFG** (0x40)
- #define **SYSCFG0_IRAWSTAT** (0xE0)
- #define **SYSCFG0_IENSTAT** (0xE4)

- #define **SYSCFG0_IENSET** (0xE8)
- #define **SYSCFG0_IENCLR** (0xEC)
- #define **SYSCFG0_EOI** (0xF0)
- #define **SYSCFG0_FLTADDRR** (0xF4)
- #define **SYSCFG0_FLTSTAT** (0xF8)
- #define **SYSCFG0_MSTPRI0** (0x110)
- #define **SYSCFG0_MSTPRI1** (0x114)
- #define **SYSCFG0_MSTPRI2** (0x118)
- #define **SYSCFG0_PINMUX**(n) (0x120 + (n ∗ 4))
- #define **SYSCFG0_SUSPSRC** (0x170)
- #define **SYSCFG0_CHIPSIG** (0x174)
- #define **SYSCFG0_CHIPSIG_CLR** (0x178)
- #define **SYSCFG0_CFGCHIP0** (0x17C)
- #define **SYSCFG0_CFGCHIP1** (0x180)
- #define **SYSCFG0_CFGCHIP2** (0x184)
- #define **SYSCFG0_CFGCHIP3** (0x188)
- #define **SYSCFG0_CFGCHIP4** (0x18C)
- #define **SYSCFG_REVID_REVID** (0xFFFFFFFFu)
- #define **SYSCFG_REVID_REVID_SHIFT** (0x00000000u)
- #define **SYSCFG_DIEIDR0_DIEID0** (0xFFFFFFFFu)
- #define **SYSCFG_DIEIDR0_DIEID0_SHIFT** (0x00000000u)
- #define **SYSCFG_DIEIDR1_DIEID1** (0xFFFFFFFFu)
- #define **SYSCFG_DIEIDR1_DIEID1_SHIFT** (0x00000000u)
- #define **SYSCFG_DIEIDR2_DIEID2** (0xFFFFFFFFu)
- #define **SYSCFG_DIEIDR2_DIEID2_SHIFT** (0x00000000u)
- #define **SYSCFG_DIEIDR3_DIEID3** (0xFFFFFFFFu)
- #define **SYSCFG_DIEIDR3_DIEID3_SHIFT** (0x00000000u)
- #define **SYSCFG_DEVIDR0_DEVID0** (0xFFFFFFFFu)
- #define **SYSCFG_DEVIDR0_DEVID0_SHIFT** (0x00000000u)
- #define **SYSCFG_BOOTCFG_SMARTRFLX** (0x0FFF0000u)
- #define **SYSCFG_BOOTCFG_SMARTRFLX_SHIFT** (0x00000010u)
- #define **SYSCFG_BOOTCFG_BOOTMODE** (0x0000FFFFu)
- #define **SYSCFG_BOOTCFG_BOOTMODE_SHIFT** (0x00000000u)
- #define **SYSCFG_CHIPREVIDR_CHIPREVID** (0x0000003Fu)
- #define **SYSCFG_CHIPREVIDR_CHIPREVID_SHIFT** (0x00000000u)
- #define **SYSCFG_KICK0R_KICK0** (0xFFFFFFFFu)
- #define **SYSCFG_KICK0R_KICK0_SHIFT** (0x00000000u)
- #define **SYSCFG_KICK0R_UNLOCK** (0x83E70B13u)
- #define **SYSCFG_KICK1R_KICK1** (0xFFFFFFFFu)
- #define **SYSCFG_KICK1R_KICK1_SHIFT** (0x00000000u)
- #define **SYSCFG_KICK1R_UNLOCK** (0x95A4F1E0u)
- #define **SYSCFG_HOST0CFG_BOOTRDY** (0x80000000u)
- #define **SYSCFG_HOST0CFG_BOOTRDY_SHIFT** (0x0000001Fu)
- #define **SYSCFG_HOST1CFG_BOOTRDY** (0x80000000u)
- #define **SYSCFG_HOST1CFG_BOOTRDY_SHIFT** (0x0000001Fu)
- #define **SYSCFG_HOST1CFG_DSP_ISTP_RST_VAL** (0x003FFFFFu)
- #define **SYSCFG_HOST1CFG_DSP_ISTP_RST_VAL_SHIFT** (0x00000000u)
- #define **SYSCFG_IRAWSTAT_ADDRERR** (0x00000002u)
- #define **SYSCFG_IRAWSTAT_ADDRERR_SHIFT** (0x00000001u)
- #define **SYSCFG_IRAWSTAT_PROTERR** (0x00000001u)
- #define **SYSCFG_IRAWSTAT_PROTERR_SHIFT** (0x00000000u)
- #define **SYSCFG_IENSTAT_ADDRERR** (0x00000002u)
- #define **SYSCFG_IENSTAT_ADDRERR_SHIFT** (0x00000001u)
- #define **SYSCFG_IENSTAT_PROTERR** (0x00000001u)
- #define **SYSCFG_IENSTAT_PROTERR_SHIFT** (0x00000000u)

- #define **SYSCFG_IENSET_ADDRERR_EN** (0x00000002u)
- #define **SYSCFG_IENSET_ADDRERR_EN_SHIFT** (0x00000001u)
- #define **SYSCFG_IENSET_PROTERR_EN** (0x00000001u)
- #define **SYSCFG_IENSET_PROTERR_EN_SHIFT** (0x00000000u)
- #define **SYSCFG_IENCLR_ADDRERR_CLR** (0x00000002u)
- #define **SYSCFG_IENCLR_ADDRERR_CLR_SHIFT** (0x00000001u)
- #define **SYSCFG_IENCLR_PROTERR_CLR** (0x00000001u)
- #define **SYSCFG_IENCLR_PROTERR_CLR_SHIFT** (0x00000000u)
- #define **SYSCFG_EOI_EOIVECT** (0x000000FFu)
- #define **SYSCFG_EOI_EOIVECT_SHIFT** (0x00000000u)
- #define **SYSCFG_FLTADDRR_FLTADDR** (0xFFFFFFFFu)
- #define **SYSCFG_FLTADDRR_FLTADDR_SHIFT** (0x00000000u)
- #define **SYSCFG_FLTSTAT_ID** (0xFF000000u)
- #define **SYSCFG_FLTSTAT_ID_SHIFT** (0x00000018u)
- #define **SYSCFG_FLTSTAT_MSTID** (0x00FF0000u)
- #define **SYSCFG_FLTSTAT_MSTID_SHIFT** (0x00000010u)
- #define **SYSCFG_FLTSTAT_PRIVID** (0x00001E00u)
- #define **SYSCFG_FLTSTAT_PRIVID_SHIFT** (0x00000009u)
- #define **SYSCFG_FLTSTAT_NOSECACC** (0x00000080u)
- #define **SYSCFG_FLTSTAT_NOSECACC_SHIFT** (0x00000007u)
- #define **SYSCFG_FLTSTAT_TYPE** (0x0000003Fu)
- #define **SYSCFG_FLTSTAT_TYPE_SHIFT** (0x00000000u)
- #define **SYSCFG_FLTSTAT_TYPE_NOFLT** (0x00000000u)
- #define **SYSCFG_FLTSTAT_TYPE_USREXE** (0x00000001u)
- #define **SYSCFG_FLTSTAT_TYPE_USRWR** (0x00000002u)
- #define **SYSCFG_FLTSTAT_TYPE_USRRD** (0x00000004u)
- #define **SYSCFG_FLTSTAT_TYPE_SPREXE** (0x00000008u)
- #define **SYSCFG_FLTSTAT_TYPE_SPRWR** (0x00000010u)
- #define **SYSCFG_FLTSTAT_TYPE_SPRRD** (0x00000020u)
- #define **SYSCFG_MSTPRI0_SATA** (0x00700000u)
- #define **SYSCFG_MSTPRI0_SATA_SHIFT** (0x00000014u)
- #define **SYSCFG_MSTPRI0_UPP** (0x00070000u)
- #define **SYSCFG_MSTPRI0_UPP_SHIFT** (0x00000010u)
- #define **SYSCFG_MSTPRI0_DSP_CFG** (0x00007000u)
- #define **SYSCFG_MSTPRI0_DSP_CFG_SHIFT** (0x0000000Cu)
- #define **SYSCFG_MSTPRI0_DSP_MDMA** (0x00000700u)
- #define **SYSCFG_MSTPRI0_DSP_MDMA_SHIFT** (0x00000008u)
- #define **SYSCFG_MSTPRI0_ARM_D** (0x00000070u)
- #define **SYSCFG_MSTPRI0_ARM_D_SHIFT** (0x00000004u)
- #define **SYSCFG_MSTPRI0_ARM_I** (0x00000007u)
- #define **SYSCFG_MSTPRI0_ARM_I_SHIFT** (0x00000000u)
- #define **SYSCFG_MSTPRI1_VPIF_DMA_1** (0x70000000u)
- #define **SYSCFG_MSTPRI1_VPIF_DMA_1_SHIFT** (0x0000001Cu)
- #define **SYSCFG_MSTPRI1_VPIF_DMA_0** (0x07000000u)
- #define **SYSCFG_MSTPRI1_VPIF_DMA_0_SHIFT** (0x00000018u)
- #define **SYSCFG_MSTPRI1_EDMA31TC0** (0x00070000u)
- #define **SYSCFG_MSTPRI1_EDMA31TC0_SHIFT** (0x00000010u)
- #define **SYSCFG_MSTPRI1_EDMA30TC1** (0x00007000u)
- #define **SYSCFG_MSTPRI1_EDMA30TC1_SHIFT** (0x0000000Cu)
- #define **SYSCFG_MSTPRI1_EDMA30TC0** (0x00000700u)
- #define **SYSCFG_MSTPRI1_EDMA30TC0_SHIFT** (0x00000008u)
- #define **SYSCFG_MSTPRI1_PRU1** (0x00000070u)
- #define **SYSCFG_MSTPRI1_PRU1_SHIFT** (0x00000004u)
- #define **SYSCFG_MSTPRI1_PRU0** (0x00000007u)
- #define **SYSCFG_MSTPRI1_PRU0_SHIFT** (0x00000000u)

- #define **SYSCFG_MSTPRI2_LCDC** (0x70000000u)
- #define **SYSCFG_MSTPRI2_LCDC_SHIFT** (0x0000001Cu)
- #define **SYSCFG_MSTPRI2_USB1** (0x07000000u)
- #define **SYSCFG_MSTPRI2_USB1_SHIFT** (0x00000018u)
- #define **SYSCFG_MSTPRI2_UHPI** (0x00700000u)
- #define **SYSCFG_MSTPRI2_UHPI_SHIFT** (0x00000014u)
- #define **SYSCFG_MSTPRI2_USB0CDMA** (0x00007000u)
- #define **SYSCFG_MSTPRI2_USB0CDMA_SHIFT** (0x0000000Cu)
- #define **SYSCFG_MSTPRI2_USB0CFG** (0x00000700u)
- #define **SYSCFG_MSTPRI2_USB0CFG_SHIFT** (0x00000008u)
- #define **SYSCFG_MSTPRI2_EMAC** (0x00000007u)
- #define **SYSCFG_MSTPRI2_EMAC_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX0_PINMUX0_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_31_28_RESERVED1** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_31_28_ALARM** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_31_28_UART2_CTS** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_31_28_GPIO0_8** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX0_PINMUX0_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_27_24_AMUTE0** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_27_24_PRU0_R30_16** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_27_24_UART2_RTS** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_27_24_GPIO0_9** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX0_PINMUX0_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX0_PINMUX0_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_23_20_AHCLKX0** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_23_20_USB_REFCLKIN** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_23_20_UART1_CTS** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_23_20_GPIO0_10** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX0_PINMUX0_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX0_PINMUX0_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_19_16_AHCLKR0** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_19_16_PRU0_R30_18** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_19_16_UART1_RTS** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_19_16_GPIO0_11** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX0_PINMUX0_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX0_PINMUX0_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_15_12_AFSX0** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_15_12_OBSERVE0_LOS** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_15_12_GPIO0_12** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX0_PINMUX0_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_11_8_AFSR0** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_11_8_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_11_8_OBSERVE0_SYNC** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_11_8_GPIO0_13** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_7_4** (0x000000F0u)

- #define **SYSCFG_PINMUX0_PINMUX0_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_7_4_ACLKX0** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_7_4_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_7_4_PRU0_R30_19** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_7_4_GPIO0_14** (0x00000008u)
- #define **SYSCFG_PINMUX0_PINMUX0_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX0_PINMUX0_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX0_PINMUX0_3_0_ACLKR0** (0x00000001u)
- #define **SYSCFG_PINMUX0_PINMUX0_3_0_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX0_PINMUX0_3_0_PRU0_R30_20** (0x00000004u)
- #define **SYSCFG_PINMUX0_PINMUX0_3_0_GPIO0_15** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX1_PINMUX1_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_31_28_AXR0_8** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_31_28_CLKS1** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_31_28_ECAP1** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_31_28_GPIO0_0** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX1_PINMUX1_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_27_24_AXR0_9** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_27_24_DX1** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_27_24_OBSERVE0_PHY_STATE2** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_27_24_GPIO0_1** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX1_PINMUX1_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX1_PINMUX1_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_23_20_AXR0_10** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_23_20_DR1** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_23_20_OBSERVE0_PHY_STATE1** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_23_20_GPIO0_2** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX1_PINMUX1_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX1_PINMUX1_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_19_16_AXR0_11** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_19_16_FSX1** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_19_16_OBSERVE0_PHY_STATE0** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_19_16_GPIO0_3** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX1_PINMUX1_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX1_PINMUX1_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_15_12_AXR0_12** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_15_12_FSR1** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_15_12_OBSERVE0_PHY_READY** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_15_12_GPIO0_4** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX1_PINMUX1_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_11_8_AXR0_13** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_11_8_CLKX1** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_11_8_OBSERVE0_COMINIT** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_11_8_GPIO0_5** (0x00000008u)

- #define **SYSCFG_PINMUX1_PINMUX1_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX1_PINMUX1_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_7_4_AXR0_14** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_7_4_CLKR1** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_7_4_OBSERVE0_COMWAKE** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_7_4_GPIO0_6** (0x00000008u)
- #define **SYSCFG_PINMUX1_PINMUX1_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX1_PINMUX1_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX1_PINMUX1_3_0_AXR0_15** (0x00000001u)
- #define **SYSCFG_PINMUX1_PINMUX1_3_0_EPWM0TZ0** (0x00000002u)
- #define **SYSCFG_PINMUX1_PINMUX1_3_0_ECAP2** (0x00000004u)
- #define **SYSCFG_PINMUX1_PINMUX1_3_0_GPIO0_7** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX2_PINMUX2_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_31_28_AXR0_0** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_31_28_ECAP0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_31_28_GPIO8_7** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_31_28_MII_TXD0** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX2_PINMUX2_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_27_24_AXR0_1** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_27_24_DX0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_27_24_GPIO1_9** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_27_24_MII_TXD1** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX2_PINMUX2_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX2_PINMUX2_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_23_20_AXR0_2** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_23_20_DR0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_23_20_GPIO1_10** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_23_20_MII_TXD2** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX2_PINMUX2_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX2_PINMUX2_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_19_16_AXR0_3** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_19_16_FSX0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_19_16_GPIO1_11** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_19_16_MII_TXD3** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX2_PINMUX2_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX2_PINMUX2_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_15_12_AXR0_4** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_15_12_FSR0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_15_12_GPIO1_12** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_15_12_MII_COL** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX2_PINMUX2_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_11_8_AXR0_5** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_11_8_CLKX0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_11_8_GPIO1_13** (0x00000004u)

- #define **SYSCFG_PINMUX2_PINMUX2_11_8_MII_TXCLK** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX2_PINMUX2_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_7_4_AXR0_6** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_7_4_CLKR0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_7_4_GPIO1_14** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_7_4_MII_TXEN** (0x00000008u)
- #define **SYSCFG_PINMUX2_PINMUX2_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX2_PINMUX2_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX2_PINMUX2_3_0_AXR0_7** (0x00000001u)
- #define **SYSCFG_PINMUX2_PINMUX2_3_0_EPWM1TZ0** (0x00000002u)
- #define **SYSCFG_PINMUX2_PINMUX2_3_0_PRU0_R30_17** (0x00000004u)
- #define **SYSCFG_PINMUX2_PINMUX2_3_0_GPIO1_15** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX3_PINMUX3_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_31_28_NSPI0_SCS2** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_31_28_UART0_RTS** (0x00000002u)
- #define **SYSCFG_PINMUX3_PINMUX3_31_28_GPIO8_1** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_31_28_MII_RXD0** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX3_PINMUX3_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_27_24_NSPI0_SCS3** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_27_24_UART0_CTS** (0x00000002u)
- #define **SYSCFG_PINMUX3_PINMUX3_27_24_GPIO8_2** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_27_24_MII_RXD1** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX3_PINMUX3_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX3_PINMUX3_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_23_20_NSPI0_SCS4** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_23_20_UART0_TXD** (0x00000002u)
- #define **SYSCFG_PINMUX3_PINMUX3_23_20_GPIO8_3** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_23_20_MII_RXD2** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX3_PINMUX3_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX3_PINMUX3_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_19_16_NSPI0_SCS5** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_19_16_UART0_RXD** (0x00000002u)
- #define **SYSCFG_PINMUX3_PINMUX3_19_16_GPIO8_4** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_19_16_MII_RXD3** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX3_PINMUX3_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX3_PINMUX3_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_15_12_SPI0_SIMO0** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_15_12_EPWMSYNCO** (0x00000002u)
- #define **SYSCFG_PINMUX3_PINMUX3_15_12_GPIO8_5** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_15_12_MII_CRS** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX3_PINMUX3_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_11_8_SPI0_SOMI0** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_11_8_EPWMSYNCI** (0x00000002u)

- #define **SYSCFG_PINMUX3_PINMUX3_11_8_GPIO8_6** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_11_8_MII_RXER** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX3_PINMUX3_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_7_4_NSPI0_ENA** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_7_4_EPWM0B** (0x00000002u)
- #define **SYSCFG_PINMUX3_PINMUX3_7_4_PRU0_R30_6** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_7_4_MII_RXDV** (0x00000008u)
- #define **SYSCFG_PINMUX3_PINMUX3_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX3_PINMUX3_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX3_PINMUX3_3_0_SPI0_CLK** (0x00000001u)
- #define **SYSCFG_PINMUX3_PINMUX3_3_0_EPWM0A** (0x00000002u)
- #define **SYSCFG_PINMUX3_PINMUX3_3_0_GPIO1_8** (0x00000004u)
- #define **SYSCFG_PINMUX3_PINMUX3_3_0_MII_RXCLK** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX4_PINMUX4_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_31_28_NSPI1_SCS2** (0x00000001u)
- #define **SYSCFG_PINMUX4_PINMUX4_31_28_UART1_TXD** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_31_28_CP_POD** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_31_28_GPIO1_0** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX4_PINMUX4_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_27_24_NSPI1_SCS3** (0x00000001u)
- #define **SYSCFG_PINMUX4_PINMUX4_27_24_UART1_RXD** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_27_24_LED** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_27_24_GPIO1_1** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX4_PINMUX4_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX4_PINMUX4_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_23_20_NSPI1_SCS4** (0x00000001u)
- #define **SYSCFG_PINMUX4_PINMUX4_23_20_UART2_TXD** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_23_20_I2C1_SDA** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_23_20_GPIO1_2** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX4_PINMUX4_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX4_PINMUX4_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_19_16_NSPI1_SCS5** (0x00000001u)
- #define **SYSCFG_PINMUX4_PINMUX4_19_16_UART2_RXD** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_19_16_I2C1_SCL** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_19_16_GPIO1_3** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX4_PINMUX4_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX4_PINMUX4_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_15_12_NSPI1_SCS6** (0x00000001u)
- #define **SYSCFG_PINMUX4_PINMUX4_15_12_I2C0_SDA** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_15_12_TM64P3_OUT12** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_15_12_GPIO1_4** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX4_PINMUX4_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_11_8_NSPI1_SCS7** (0x00000001u)

- #define **SYSCFG_PINMUX4_PINMUX4_11_8_I2C0_SCL** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_11_8_TM64P2_OUT12** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_11_8_GPIO1_5** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX4_PINMUX4_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_7_4_NSPI0_SCS0** (0x00000001u)
- #define **SYSCFG_PINMUX4_PINMUX4_7_4_TM64P1_OUT12** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_7_4_GPIO1_6** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_7_4_MDIO_D** (0x00000008u)
- #define **SYSCFG_PINMUX4_PINMUX4_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX4_PINMUX4_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX4_PINMUX4_3_0_NSPI0_SCS1** (0x00000001u)
- #define **SYSCFG_PINMUX4_PINMUX4_3_0_TM64P0_OUT12** (0x00000002u)
- #define **SYSCFG_PINMUX4_PINMUX4_3_0_GPIO1_7** (0x00000004u)
- #define **SYSCFG_PINMUX4_PINMUX4_3_0_MDIO_CLK** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX5_PINMUX5_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_31_28_EMA_BA0** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_31_28_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_31_28_GPIO2_8** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX5_PINMUX5_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_27_24_EMA_BA1** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_27_24_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_27_24_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_27_24_GPIO2_9** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX5_PINMUX5_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX5_PINMUX5_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_23_20_SPI1_SIMO0** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_23_20_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_23_20_GPIO2_10** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX5_PINMUX5_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX5_PINMUX5_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_19_16_SPI1_SOMI0** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_19_16_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_19_16_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_19_16_GPIO2_11** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX5_PINMUX5_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX5_PINMUX5_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_15_12_NSPI1_ENA** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_15_12_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_15_12_GPIO2_12** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX5_PINMUX5_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_11_8_DEFAULT** (0x00000000u)

- #define **SYSCFG_PINMUX5_PINMUX5_11_8_SPI1_CLK** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_11_8_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_11_8_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_11_8_GPIO2_13** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX5_PINMUX5_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_7_4_NSPI1_SCS0** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_7_4_EPWM1B** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_7_4_PRU0_R30_7** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_7_4_GPIO2_14** (0x00000008u)
- #define **SYSCFG_PINMUX5_PINMUX5_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX5_PINMUX5_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX5_PINMUX5_3_0_NSPI1_SCS1** (0x00000001u)
- #define **SYSCFG_PINMUX5_PINMUX5_3_0_EPWM1A** (0x00000002u)
- #define **SYSCFG_PINMUX5_PINMUX5_3_0_PRU0_R30_8** (0x00000004u)
- #define **SYSCFG_PINMUX5_PINMUX5_3_0_GPIO2_15** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX6_PINMUX6_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_31_28_NEMA_CS0** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_31_28_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_31_28_GPIO2_0** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX6_PINMUX6_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_27_24_EMA_WAIT1** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_27_24_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_27_24_PRU0_R30_1** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_27_24_GPIO2_1** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX6_PINMUX6_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX6_PINMUX6_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_23_20_NEMA_WE_DQM1** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_23_20_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_23_20_GPIO2_2** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX6_PINMUX6_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX6_PINMUX6_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_19_16_NEMA_WE_DQM0** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_19_16_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_19_16_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_19_16_GPIO2_3** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX6_PINMUX6_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX6_PINMUX6_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_15_12_NEMA_CAS** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_15_12_PRU0_R30_2** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_15_12_GPIO2_4** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX6_PINMUX6_11_8_SHIFT** (0x00000008u)

- #define **SYSCFG_PINMUX6_PINMUX6_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_11_8_NEMA_RAS** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_11_8_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_11_8_PRU0_R30_3** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_11_8_GPIO2_5** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX6_PINMUX6_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_7_4_EMA_SDCKE** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_7_4_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_7_4_PRU0_R30_4** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_7_4_GPIO2_6** (0x00000008u)
- #define **SYSCFG_PINMUX6_PINMUX6_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX6_PINMUX6_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX6_PINMUX6_3_0_EMA_CLK** (0x00000001u)
- #define **SYSCFG_PINMUX6_PINMUX6_3_0_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX6_PINMUX6_3_0_PRU0_R30_5** (0x00000004u)
- #define **SYSCFG_PINMUX6_PINMUX6_3_0_GPIO2_7** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX7_PINMUX7_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_31_28_EMA_WAIT0** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_31_28_PRU0_R30_0** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_31_28_GPIO3_8** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX7_PINMUX7_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_27_24_NEMA_RNW** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_27_24_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_27_24_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_27_24_GPIO3_9** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX7_PINMUX7_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX7_PINMUX7_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_23_20_NEMA_OE** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_23_20_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_23_20_GPIO3_10** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX7_PINMUX7_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX7_PINMUX7_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_19_16_NEMA_WE** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_19_16_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_19_16_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_19_16_GPIO3_11** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX7_PINMUX7_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX7_PINMUX7_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_15_12_NEMA_CS5** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_15_12_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_15_12_GPIO3_12** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_11_8** (0x00000F00u)

- #define **SYSCFG_PINMUX7_PINMUX7_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_11_8_NEMA_CS4** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_11_8_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_11_8_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_11_8_GPIO3_13** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX7_PINMUX7_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_7_4_NEMA_CS3** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_7_4_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_7_4_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_7_4_GPIO3_14** (0x00000008u)
- #define **SYSCFG_PINMUX7_PINMUX7_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX7_PINMUX7_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX7_PINMUX7_3_0_NEMA_CS2** (0x00000001u)
- #define **SYSCFG_PINMUX7_PINMUX7_3_0_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX7_PINMUX7_3_0_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX7_PINMUX7_3_0_GPIO3_15** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX8_PINMUX8_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_31_28_EMA_D8** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_31_28_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_31_28_GPIO3_0** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX8_PINMUX8_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_27_24_EMA_D9** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_27_24_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_27_24_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_27_24_GPIO3_1** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX8_PINMUX8_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX8_PINMUX8_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_23_20_EMA_D10** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_23_20_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_23_20_GPIO3_2** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX8_PINMUX8_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX8_PINMUX8_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_19_16_EMA_D11** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_19_16_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_19_16_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_19_16_GPIO3_3** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX8_PINMUX8_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX8_PINMUX8_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_15_12_EMA_D12** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_15_12_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_15_12_GPIO3_4** (0x00000008u)

- #define **SYSCFG_PINMUX8_PINMUX8_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX8_PINMUX8_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_11_8_EMA_D13** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_11_8_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_11_8_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_11_8_GPIO3_5** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX8_PINMUX8_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_7_4_EMA_D14** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_7_4_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_7_4_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_7_4_GPIO3_6** (0x00000008u)
- #define **SYSCFG_PINMUX8_PINMUX8_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX8_PINMUX8_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX8_PINMUX8_3_0_EMA_D15** (0x00000001u)
- #define **SYSCFG_PINMUX8_PINMUX8_3_0_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX8_PINMUX8_3_0_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX8_PINMUX8_3_0_GPIO3_7** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX9_PINMUX9_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_31_28_EMA_D0** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_31_28_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_31_28_GPIO4_8** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX9_PINMUX9_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_27_24_EMA_D1** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_27_24_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_27_24_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_27_24_GPIO4_9** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX9_PINMUX9_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX9_PINMUX9_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_23_20_EMA_D2** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_23_20_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_23_20_GPIO4_10** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX9_PINMUX9_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX9_PINMUX9_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_19_16_EMA_D3** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_19_16_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_19_16_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_19_16_GPIO4_11** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX9_PINMUX9_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX9_PINMUX9_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_15_12_EMA_D4** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_15_12_RESERVED4** (0x00000004u)

- #define **SYSCFG_PINMUX9_PINMUX9_15_12_GPIO4_12** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX9_PINMUX9_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_11_8_EMA_D5** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_11_8_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_11_8_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_11_8_GPIO4_13** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX9_PINMUX9_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_7_4_EMA_D6** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_7_4_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_7_4_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_7_4_GPIO4_14** (0x00000008u)
- #define **SYSCFG_PINMUX9_PINMUX9_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX9_PINMUX9_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX9_PINMUX9_3_0_EMA_D7** (0x00000001u)
- #define **SYSCFG_PINMUX9_PINMUX9_3_0_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX9_PINMUX9_3_0_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX9_PINMUX9_3_0_GPIO4_15** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX10_PINMUX10_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_31_28_EMA_A16** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_31_28_MMCSD0_DAT5** (0x00000002u)
- #define **SYSCFG_PINMUX10_PINMUX10_31_28_PRU1_R30_24** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_31_28_GPIO4_0** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX10_PINMUX10_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_27_24_EMA_A17** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_27_24_MMCSD0_DAT4** (0x00000002u)
- #define **SYSCFG_PINMUX10_PINMUX10_27_24_PRU1_R30_25** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_27_24_GPIO4_1** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX10_PINMUX10_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX10_PINMUX10_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_23_20_EMA_A18** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_23_20_MMCSD0_DAT3** (0x00000002u)
- #define **SYSCFG_PINMUX10_PINMUX10_23_20_PRU1_R30_26** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_23_20_GPIO4_2** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX10_PINMUX10_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX10_PINMUX10_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_19_16_EMA_A19** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_19_16_MMCSD0_DAT2** (0x00000002u)
- #define **SYSCFG_PINMUX10_PINMUX10_19_16_PRU1_R30_27** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_19_16_GPIO4_3** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX10_PINMUX10_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX10_PINMUX10_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_15_12_EMA_A20** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_15_12_MMCSD0_DAT1** (0x00000002u)

- #define **SYSCFG_PINMUX10_PINMUX10_15_12_PRU1_R30_28** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_15_12_GPIO4_4** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX10_PINMUX10_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_11_8_EMA_A21** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_11_8_MMCSD0_DAT0** (0x00000002u)
- #define **SYSCFG_PINMUX10_PINMUX10_11_8_PRU1_R30_29** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_11_8_GPIO4_5** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX10_PINMUX10_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_7_4_EMA_A22** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_7_4_MMCSD0_CMD** (0x00000002u)
- #define **SYSCFG_PINMUX10_PINMUX10_7_4_PRU1_R30_30** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_7_4_GPIO4_6** (0x00000008u)
- #define **SYSCFG_PINMUX10_PINMUX10_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX10_PINMUX10_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX10_PINMUX10_3_0_EMA_A23** (0x00000001u)
- #define **SYSCFG_PINMUX10_PINMUX10_3_0_MMCSD0_CLK** (0x00000002u)
- #define **SYSCFG_PINMUX10_PINMUX10_3_0_PRU1_R30_31** (0x00000004u)
- #define **SYSCFG_PINMUX10_PINMUX10_3_0_GPIO4_7** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX11_PINMUX11_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_31_28_EMA_A8** (0x00000001u)
- #define **SYSCFG_PINMUX11_PINMUX11_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_31_28_PRU1_R30_16** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_31_28_GPIO5_8** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX11_PINMUX11_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_27_24_EMA_A9** (0x00000001u)
- #define **SYSCFG_PINMUX11_PINMUX11_27_24_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_27_24_PRU1_R30_17** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_27_24_GPIO5_9** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX11_PINMUX11_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX11_PINMUX11_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_23_20_EMA_A10** (0x00000001u)
- #define **SYSCFG_PINMUX11_PINMUX11_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_23_20_PRU1_R30_18** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_23_20_GPIO5_10** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX11_PINMUX11_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX11_PINMUX11_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_19_16_EMA_A11** (0x00000001u)
- #define **SYSCFG_PINMUX11_PINMUX11_19_16_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_19_16_PRU1_R30_19** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_19_16_GPIO5_11** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX11_PINMUX11_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX11_PINMUX11_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_15_12_EMA_A12** (0x00000001u)

- #define **SYSCFG_PINMUX11_PINMUX11_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_15_12_PRU1_R30_20** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_15_12_GPIO5_12** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX11_PINMUX11_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_11_8_EMA_A13** (0x00000001u)
- #define **SYSCFG_PINMUX11_PINMUX11_11_8_PRU0_R30_21** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_11_8_PRU1_R30_21** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_11_8_GPIO5_13** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX11_PINMUX11_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_7_4_EMA_A14** (0x00000001u)
- #define **SYSCFG_PINMUX11_PINMUX11_7_4_MMCSD0_DAT7** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_7_4_PRU1_R30_22** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_7_4_GPIO5_14** (0x00000008u)
- #define **SYSCFG_PINMUX11_PINMUX11_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX11_PINMUX11_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX11_PINMUX11_3_0_EMA_A15** (0x00000001u)
- #define **SYSCFG_PINMUX11_PINMUX11_3_0_MMCSD0_DAT6** (0x00000002u)
- #define **SYSCFG_PINMUX11_PINMUX11_3_0_PRU1_R30_23** (0x00000004u)
- #define **SYSCFG_PINMUX11_PINMUX11_3_0_GPIO5_15** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX12_PINMUX12_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_31_28_EMA_A0** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_31_28_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_31_28_GPIO5_0** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX12_PINMUX12_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_27_24_EMA_A1** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_27_24_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_27_24_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_27_24_GPIO5_1** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX12_PINMUX12_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX12_PINMUX12_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_23_20_EMA_A2** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_23_20_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_23_20_GPIO5_2** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX12_PINMUX12_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX12_PINMUX12_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_19_16_EMA_A3** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_19_16_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_19_16_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_19_16_GPIO5_3** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX12_PINMUX12_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX12_PINMUX12_15_12_DEFAULT** (0x00000000u)

- #define **SYSCFG_PINMUX12_PINMUX12_15_12_EMA_A4** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_15_12_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_15_12_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_15_12_GPIO5_4** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX12_PINMUX12_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_11_8_EMA_A5** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_11_8_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_11_8_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_11_8_GPIO5_5** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX12_PINMUX12_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_7_4_EMA_A6** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_7_4_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_7_4_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_7_4_GPIO5_6** (0x00000008u)
- #define **SYSCFG_PINMUX12_PINMUX12_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX12_PINMUX12_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX12_PINMUX12_3_0_EMA_A7** (0x00000001u)
- #define **SYSCFG_PINMUX12_PINMUX12_3_0_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX12_PINMUX12_3_0_PRU1_R30_15** (0x00000004u)
- #define **SYSCFG_PINMUX12_PINMUX12_3_0_GPIO5_7** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX13_PINMUX13_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_31_28_PRU0_R30_26** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_31_28_UHPI_HRNW** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_31_28_CH1_WAIT** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_31_28_GPIO6_8** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX13_PINMUX13_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_27_24_PRU0_R30_27** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_27_24_UHPI_HHWIL** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_27_24_GPIO6_9** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX13_PINMUX13_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX13_PINMUX13_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_23_20_PRU0_R30_28** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_23_20_UHPI_HCNTL1** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_23_20_CH1_START** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_23_20_GPIO6_10** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX13_PINMUX13_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX13_PINMUX13_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_19_16_PRU0_R30_29** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_19_16_UHPI_HCNTL0** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_19_16_CH1_CLK** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_19_16_GPIO6_11** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX13_PINMUX13_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX13_PINMUX13_15_12_DEFAULT** (0x00000000u)

- #define **SYSCFG_PINMUX13_PINMUX13_15_12_PRU0_R30_30** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_15_12_NUHPI_HINT** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_15_12_PRU1_R30_11** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_15_12_GPIO6_12** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX13_PINMUX13_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_11_8_PRU0_R30_31** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_11_8_NUHPI_HRDY** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_11_8_PRU1_R30_12** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_11_8_GPIO6_13** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX13_PINMUX13_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_7_4_OBSCLK0** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_7_4_NUHPI_HDS2** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_7_4_PRU1_R30_13** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_7_4_GPIO6_14** (0x00000008u)
- #define **SYSCFG_PINMUX13_PINMUX13_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX13_PINMUX13_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX13_PINMUX13_3_0_NRESETOUT** (0x00000001u)
- #define **SYSCFG_PINMUX13_PINMUX13_3_0_NUHPI_HAS** (0x00000002u)
- #define **SYSCFG_PINMUX13_PINMUX13_3_0_PRU1_R30_14** (0x00000004u)
- #define **SYSCFG_PINMUX13_PINMUX13_3_0_GPIO6_15** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX14_PINMUX14_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_31_28_DIN2** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_31_28_UHPI_HD10** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_31_28_UPP_D10** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_31_28_RMII_RXER** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX14_PINMUX14_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_27_24_DIN3** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_27_24_UHPI_HD11** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_27_24_UPP_D11** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_27_24_RMII_RXD0** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX14_PINMUX14_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX14_PINMUX14_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_23_20_DIN4** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_23_20_UHPI_HD12** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_23_20_UPP_D12** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_23_20_RMII_RXD1** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX14_PINMUX14_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX14_PINMUX14_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_19_16_DIN5** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_19_16_UHPI_HD13** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_19_16_UPP_D13** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_19_16_RMII_TXEN** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX14_PINMUX14_15_12_SHIFT** (0x0000000Cu)

- #define **SYSCFG_PINMUX14_PINMUX14_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_15_12_DIN6** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_15_12_UHPI_HD14** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_15_12_UPP_D14** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_15_12_RMII_TXD0** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX14_PINMUX14_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_11_8_DIN7** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_11_8_UHPI_HD15** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_11_8_UPP_D15** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_11_8_RMII_TXD1** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX14_PINMUX14_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_7_4_CLKIN1** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_7_4_NUHPI_HDS1** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_7_4_PRU1_R30_9** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_7_4_GPIO6_6** (0x00000008u)
- #define **SYSCFG_PINMUX14_PINMUX14_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX14_PINMUX14_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX14_PINMUX14_3_0_CLKIN0** (0x00000001u)
- #define **SYSCFG_PINMUX14_PINMUX14_3_0_NUHPI_HCS** (0x00000002u)
- #define **SYSCFG_PINMUX14_PINMUX14_3_0_PRU1_R30_10** (0x00000004u)
- #define **SYSCFG_PINMUX14_PINMUX14_3_0_GPIO6_7** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX15_PINMUX15_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_31_28_DIN10** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_31_28_UHPI_HD2** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_31_28_UPP_D2** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_31_28_PRU0_R30_10** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX15_PINMUX15_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_27_24_DIN11** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_27_24_UHPI_HD3** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_27_24_UPP_D3** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_27_24_PRU0_R30_11** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX15_PINMUX15_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX15_PINMUX15_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_23_20_DIN12** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_23_20_UHPI_HD4** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_23_20_UPP_D4** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_23_20_PRU0_R30_12** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX15_PINMUX15_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX15_PINMUX15_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_19_16_DIN13_FIELD** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_19_16_UHPI_HD5** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_19_16_UPP_D5** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_19_16_PRU0_R30_13** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_15_12** (0x0000F000u)

- #define **SYSCFG_PINMUX15_PINMUX15_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX15_PINMUX15_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_15_12_DIN14_HSYNC** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_15_12_UHPI_HD6** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_15_12_UPP_D6** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_15_12_PRU0_R30_14** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX15_PINMUX15_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_11_8_DIN15_VSYNC** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_11_8_UHPI_HD7** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_11_8_UPP_D7** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_11_8_PRU0_R30_15** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX15_PINMUX15_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_7_4_DIN0** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_7_4_UHPI_HD8** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_7_4_UPP_D8** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_7_4_RMII_CRS_DV** (0x00000008u)
- #define **SYSCFG_PINMUX15_PINMUX15_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX15_PINMUX15_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX15_PINMUX15_3_0_DIN1** (0x00000001u)
- #define **SYSCFG_PINMUX15_PINMUX15_3_0_UHPI_HD9** (0x00000002u)
- #define **SYSCFG_PINMUX15_PINMUX15_3_0_UPP_D9** (0x00000004u)
- #define **SYSCFG_PINMUX15_PINMUX15_3_0_RMII_MHZ_50_CLK** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX16_PINMUX16_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_31_28_DOUT2** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_31_28_LCD_D2** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_31_28_UPP_XD10** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_31_28_GPIO7_10** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX16_PINMUX16_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_27_24_DOUT3** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_27_24_LCD_D3** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_27_24_UPP_XD11** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_27_24_GPIO7_11** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX16_PINMUX16_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX16_PINMUX16_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_23_20_DOUT4** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_23_20_LCD_D4** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_23_20_UPP_XD12** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_23_20_GPIO7_12** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX16_PINMUX16_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX16_PINMUX16_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_19_16_DOUT5** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_19_16_LCD_D5** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_19_16_UPP_XD13** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_19_16_GPIO7_13** (0x00000008u)

- #define **SYSCFG_PINMUX16_PINMUX16_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX16_PINMUX16_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX16_PINMUX16_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_15_12_DOUT6** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_15_12_LCD_D6** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_15_12_UPP_XD14** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_15_12_GPIO7_14** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX16_PINMUX16_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_11_8_DOUT7** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_11_8_LCD_D7** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_11_8_UPP_XD15** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_11_8_GPIO7_15** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX16_PINMUX16_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_7_4_DIN8** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_7_4_UHPI_HD0** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_7_4_UPP_D0** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_7_4_GPIO6_5** (0x00000008u)
- #define **SYSCFG_PINMUX16_PINMUX16_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX16_PINMUX16_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX16_PINMUX16_3_0_DIN9** (0x00000001u)
- #define **SYSCFG_PINMUX16_PINMUX16_3_0_UHPI_HD1** (0x00000002u)
- #define **SYSCFG_PINMUX16_PINMUX16_3_0_UPP_D1** (0x00000004u)
- #define **SYSCFG_PINMUX16_PINMUX16_3_0_PRU0_R30_9** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX17_PINMUX17_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_31_28_DOUT10** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_31_28_LCD_D10** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_31_28_UPP_XD2** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_31_28_GPIO7_2** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX17_PINMUX17_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_27_24_DOUT11** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_27_24_LCD_D11** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_27_24_UPP_XD3** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_27_24_GPIO7_3** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX17_PINMUX17_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX17_PINMUX17_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_23_20_DOUT12** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_23_20_LCD_D12** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_23_20_UPP_XD4** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_23_20_GPIO7_4** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX17_PINMUX17_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX17_PINMUX17_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_19_16_DOUT13** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_19_16_LCD_D13** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_19_16_UPP_XD5** (0x00000004u)

- #define **SYSCFG_PINMUX17_PINMUX17_19_16_GPIO7_5** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX17_PINMUX17_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX17_PINMUX17_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_15_12_DOUT14** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_15_12_LCD_D14** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_15_12_UPP_XD6** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_15_12_GPIO7_6** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX17_PINMUX17_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_11_8_DOUT15** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_11_8_LCD_D15** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_11_8_UPP_XD7** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_11_8_GPIO7_7** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX17_PINMUX17_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_7_4_DOUT0** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_7_4_LCD_D0** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_7_4_UPP_XD8** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_7_4_GPIO7_8** (0x00000008u)
- #define **SYSCFG_PINMUX17_PINMUX17_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX17_PINMUX17_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX17_PINMUX17_3_0_DOUT1** (0x00000001u)
- #define **SYSCFG_PINMUX17_PINMUX17_3_0_LCD_D1** (0x00000002u)
- #define **SYSCFG_PINMUX17_PINMUX17_3_0_UPP_XD9** (0x00000004u)
- #define **SYSCFG_PINMUX17_PINMUX17_3_0_GPIO7_9** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX18_PINMUX18_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_31_28_MMCSD1_DAT6** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_31_28_LCD_MCLK** (0x00000002u)
- #define **SYSCFG_PINMUX18_PINMUX18_31_28_PRU1_R30_6** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_31_28_GPIO8_10** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX18_PINMUX18_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_27_24_MMCSD1_DAT7** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_27_24_LCD_PCLK** (0x00000002u)
- #define **SYSCFG_PINMUX18_PINMUX18_27_24_PRU1_R30_7** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_27_24_GPIO8_11** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX18_PINMUX18_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX18_PINMUX18_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_23_20_PRU0_R30_22** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_23_20_PRU1_R30_8** (0x00000002u)
- #define **SYSCFG_PINMUX18_PINMUX18_23_20_CH0_WAIT** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_23_20_GPIO8_12** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX18_PINMUX18_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX18_PINMUX18_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_19_16_PRU0_R30_23** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_19_16_MMCSD1_CMD** (0x00000002u)

- #define **SYSCFG_PINMUX18_PINMUX18_19_16_GPIO8_13** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX18_PINMUX18_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX18_PINMUX18_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_15_12_PRU0_R30_24** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_15_12_MMCSD1_CLK** (0x00000002u)
- #define **SYSCFG_PINMUX18_PINMUX18_15_12_CH0_START** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_15_12_GPIO8_14** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX18_PINMUX18_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_11_8_PRU0_R30_25** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_11_8_MMCSD1_DAT0** (0x00000002u)
- #define **SYSCFG_PINMUX18_PINMUX18_11_8_CH0_CLK** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_11_8_GPIO8_15** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX18_PINMUX18_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_7_4_DOUT8** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_7_4_LCD_D8** (0x00000002u)
- #define **SYSCFG_PINMUX18_PINMUX18_7_4_UPP_XD0** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_7_4_GPIO7_0** (0x00000008u)
- #define **SYSCFG_PINMUX18_PINMUX18_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX18_PINMUX18_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX18_PINMUX18_3_0_DOUT9** (0x00000001u)
- #define **SYSCFG_PINMUX18_PINMUX18_3_0_LCD_D9** (0x00000002u)
- #define **SYSCFG_PINMUX18_PINMUX18_3_0_UPP_XD1** (0x00000004u)
- #define **SYSCFG_PINMUX18_PINMUX18_3_0_GPIO7_1** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_31_28** (0xF0000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_31_28_SHIFT** (0x0000001Cu)
- #define **SYSCFG_PINMUX19_PINMUX19_31_28_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_31_28_RTCK** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_31_28_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX19_PINMUX19_31_28_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_31_28_GPIO8_0** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_27_24** (0x0F000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_27_24_SHIFT** (0x00000018u)
- #define **SYSCFG_PINMUX19_PINMUX19_27_24_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_27_24_RESERVED1** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_27_24_NLCD_AC_ENB_CS** (0x00000002u)
- #define **SYSCFG_PINMUX19_PINMUX19_27_24_RESERVED4** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_27_24_GPIO6_0** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_23_20** (0x00F00000u)
- #define **SYSCFG_PINMUX19_PINMUX19_23_20_SHIFT** (0x00000014u)
- #define **SYSCFG_PINMUX19_PINMUX19_23_20_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_23_20_CLKO3** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_23_20_RESERVED2** (0x00000002u)
- #define **SYSCFG_PINMUX19_PINMUX19_23_20_PRU1_R30_0** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_23_20_GPIO6_1** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_19_16** (0x000F0000u)
- #define **SYSCFG_PINMUX19_PINMUX19_19_16_SHIFT** (0x00000010u)
- #define **SYSCFG_PINMUX19_PINMUX19_19_16_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_19_16_CLKIN3** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_19_16_MMCSD1_DAT1** (0x00000002u)

- #define **SYSCFG_PINMUX19_PINMUX19_19_16_PRU1_R30_1** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_19_16_GPIO6_2** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_15_12** (0x0000F000u)
- #define **SYSCFG_PINMUX19_PINMUX19_15_12_SHIFT** (0x0000000Cu)
- #define **SYSCFG_PINMUX19_PINMUX19_15_12_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_15_12_CLKO2** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_15_12_MMCSD1_DAT2** (0x00000002u)
- #define **SYSCFG_PINMUX19_PINMUX19_15_12_PRU1_R30_2** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_15_12_GPIO6_3** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_11_8** (0x00000F00u)
- #define **SYSCFG_PINMUX19_PINMUX19_11_8_SHIFT** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_11_8_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_11_8_CLKIN2** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_11_8_MMCSD1_DAT3** (0x00000002u)
- #define **SYSCFG_PINMUX19_PINMUX19_11_8_PRU1_R30_3** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_11_8_GPIO6_4** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_7_4** (0x000000F0u)
- #define **SYSCFG_PINMUX19_PINMUX19_7_4_SHIFT** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_7_4_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_7_4_MMCSD1_DAT4** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_7_4_LCD_VSYNC** (0x00000002u)
- #define **SYSCFG_PINMUX19_PINMUX19_7_4_PRU1_R30_4** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_7_4_GPIO8_8** (0x00000008u)
- #define **SYSCFG_PINMUX19_PINMUX19_3_0** (0x0000000Fu)
- #define **SYSCFG_PINMUX19_PINMUX19_3_0_SHIFT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_3_0_DEFAULT** (0x00000000u)
- #define **SYSCFG_PINMUX19_PINMUX19_3_0_MMCSD1_DAT5** (0x00000001u)
- #define **SYSCFG_PINMUX19_PINMUX19_3_0_LCD_HSYNC** (0x00000002u)
- #define **SYSCFG_PINMUX19_PINMUX19_3_0_PRU1_R30_5** (0x00000004u)
- #define **SYSCFG_PINMUX19_PINMUX19_3_0_GPIO8_9** (0x00000008u)
- #define **SYSCFG_SUSPSRC_TIMER64P_2SRC** (0x20000000u)
- #define **SYSCFG_SUSPSRC_TIMER64P_2SRC_SHIFT** (0x0000001Du)
- #define **SYSCFG_SUSPSRC_TIMER64P_1SRC** (0x10000000u)
- #define **SYSCFG_SUSPSRC_TIMER64P_1SRC_SHIFT** (0x0000001Cu)
- #define **SYSCFG_SUSPSRC_TIMER64P_0SRC** (0x08000000u)
- #define **SYSCFG_SUSPSRC_TIMER64P_0SRC_SHIFT** (0x0000001Bu)
- #define **SYSCFG_SUSPSRC_EPWM1SRC** (0x01000000u)
- #define **SYSCFG_SUSPSRC_EPWM1SRC_SHIFT** (0x00000018u)
- #define **SYSCFG_SUSPSRC_EPWM0SRC** (0x00800000u)
- #define **SYSCFG_SUSPSRC_EPWM0SRC_SHIFT** (0x00000017u)
- #define **SYSCFG_SUSPSRC_SPI1SRC** (0x00400000u)
- #define **SYSCFG_SUSPSRC_SPI1SRC_SHIFT** (0x00000016u)
- #define **SYSCFG_SUSPSRC_SPI0SRC** (0x00200000u)
- #define **SYSCFG_SUSPSRC_SPI0SRC_SHIFT** (0x00000015u)
- #define **SYSCFG_SUSPSRC_UART2SRC** (0x00100000u)
- #define **SYSCFG_SUSPSRC_UART2SRC_SHIFT** (0x00000014u)
- #define **SYSCFG_SUSPSRC_UART1SRC** (0x00080000u)
- #define **SYSCFG_SUSPSRC_UART1SRC_SHIFT** (0x00000013u)
- #define **SYSCFG_SUSPSRC_UART0SRC** (0x00040000u)
- #define **SYSCFG_SUSPSRC_UART0SRC_SHIFT** (0x00000012u)
- #define **SYSCFG_SUSPSRC_I2C1SRC** (0x00020000u)
- #define **SYSCFG_SUSPSRC_I2C1SRC_SHIFT** (0x00000011u)
- #define **SYSCFG_SUSPSRC_I2C0SRC** (0x00010000u)
- #define **SYSCFG_SUSPSRC_I2C0SRC_SHIFT** (0x00000010u)
- #define **SYSCFG_SUSPSRC_VPIFSRC** (0x00004000u)

- #define **SYSCFG_SUSPSRC_VPIFSRC_SHIFT** (0x0000000Eu)
- #define **SYSCFG_SUSPSRC_SATASRC** (0x00002000u)
- #define **SYSCFG_SUSPSRC_SATASRC_SHIFT** (0x0000000Du)
- #define **SYSCFG_SUSPSRC_HPISRC** (0x00001000u)
- #define **SYSCFG_SUSPSRC_HPISRC_SHIFT** (0x0000000Cu)
- #define **SYSCFG_SUSPSRC_USB0SRC** (0x00000200u)
- #define **SYSCFG_SUSPSRC_USB0SRC_SHIFT** (0x00000009u)
- #define **SYSCFG_SUSPSRC_MCBSP1SRC** (0x00000100u)
- #define **SYSCFG_SUSPSRC_MCBSP1SRC_SHIFT** (0x00000008u)
- #define **SYSCFG_SUSPSRC_MCBSP0SRC** (0x00000080u)
- #define **SYSCFG_SUSPSRC_MCBSP0SRC_SHIFT** (0x00000007u)
- #define **SYSCFG_SUSPSRC_PRUSRC** (0x00000040u)
- #define **SYSCFG_SUSPSRC_PRUSRC_SHIFT** (0x00000006u)
- #define **SYSCFG_SUSPSRC_EMACSRC** (0x00000020u)
- #define **SYSCFG_SUSPSRC_EMACSRC_SHIFT** (0x00000005u)
- #define **SYSCFG_SUSPSRC_UPPSRC** (0x00000010u)
- #define **SYSCFG_SUSPSRC_UPPSRC_SHIFT** (0x00000004u)
- #define **SYSCFG_SUSPSRC_TIMER64P_3SRC** (0x00000008u)
- #define **SYSCFG_SUSPSRC_TIMER64P_3SRC_SHIFT** (0x00000003u)
- #define **SYSCFG_SUSPSRC_ECAP2SRC** (0x00000004u)
- #define **SYSCFG_SUSPSRC_ECAP2SRC_SHIFT** (0x00000002u)
- #define **SYSCFG_SUSPSRC_ECAP1SRC** (0x00000002u)
- #define **SYSCFG_SUSPSRC_ECAP1SRC_SHIFT** (0x00000001u)
- #define **SYSCFG_SUSPSRC_ECAP0SRC** (0x00000001u)
- #define **SYSCFG_SUSPSRC_ECAP0SRC_SHIFT** (0x00000000u)
- #define **SYSCFG_CHIPSIG_CHIPSIG4** (0x00000010u)
- #define **SYSCFG_CHIPSIG_CHIPSIG4_SHIFT** (0x00000004u)
- #define **SYSCFG_CHIPSIG_CHIPSIG3** (0x00000008u)
- #define **SYSCFG_CHIPSIG_CHIPSIG3_SHIFT** (0x00000003u)
- #define **SYSCFG_CHIPSIG_CHIPSIG2** (0x00000004u)
- #define **SYSCFG_CHIPSIG_CHIPSIG2_SHIFT** (0x00000002u)
- #define **SYSCFG_CHIPSIG_CHIPSIG1** (0x00000002u)
- #define **SYSCFG_CHIPSIG_CHIPSIG1_SHIFT** (0x00000001u)
- #define **SYSCFG_CHIPSIG_CHIPSIG0** (0x00000001u)
- #define **SYSCFG_CHIPSIG_CHIPSIG0_SHIFT** (0x00000000u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG4** (0x00000010u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG4_SHIFT** (0x00000004u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG3** (0x00000008u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG3_SHIFT** (0x00000003u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG2** (0x00000004u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG2_SHIFT** (0x00000002u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG1** (0x00000002u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG1_SHIFT** (0x00000001u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG0** (0x00000001u)
- #define **SYSCFG_CHIPSIG_CLR_CHIPSIG0_SHIFT** (0x00000000u)
- #define **SYSCFG_CFGCHIP0_ARM_CLK_DIS0** (0x80000000u)
- #define **SYSCFG_CFGCHIP0_ARM_CLK_DIS0_SHIFT** (0x0000001Fu)
- #define **SYSCFG_CFGCHIP0_ARM_TAP_DIS0** (0x40000000u)
- #define **SYSCFG_CFGCHIP0_ARM_TAP_DIS0_SHIFT** (0x0000001Eu)
- #define **SYSCFG_CFGCHIP0_PLL_MASTER_LOCK** (0x00000010u)
- #define **SYSCFG_CFGCHIP0_PLL_MASTER_LOCK_SHIFT** (0x00000004u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC1DBS** (0x0000000Cu)
- #define **SYSCFG_CFGCHIP0_EDMA30TC1DBS_SHIFT** (0x00000002u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC1DBS_16BYTE** (0x00000000u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC1DBS_32BYTE** (0x00000001u)

- #define **SYSCFG_CFGCHIP0_EDMA30TC1DBS_64BYTE** (0x00000002u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC1DBS_RESERVED** (0x00000003u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC0DBS** (0x00000003u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC0DBS_SHIFT** (0x00000000u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC0DBS_16BYTE** (0x00000000u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC0DBS_32BYTE** (0x00000001u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC0DBS_64BYTE** (0x00000002u)
- #define **SYSCFG_CFGCHIP0_EDMA30TC0DBS_RESERVED** (0x00000003u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC** (0xF8000000u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_SHIFT** (0x0000001Bu)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_ECAP2** (0x00000000u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_MCASP0_TXDMA** (0x00000001u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_MCASP0_RXDMA** (0x00000002u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_MCASP1_TXDMA** (0x00000003u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_MCASP1_RXDMA** (0x00000004u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_MCASP2_TXDMA** (0x00000005u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_MCASP2_RXDMA** (0x00000006u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C0_RXTHPLSEINT** (0x00000007u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C0_RXPLSEINT** (0x00000008u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C0_TXPLSEINT** (0x00000009u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C0_MISCINT** (0x0000000au)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C1_RXTHPLSEINT** (0x0000000bu)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C1_RXPLSEINT** (0x0000000cu)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C1_TXPLSEINT** (0x0000000du)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C1_MISCINT** (0x0000000eu)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C2_RXTHPLSEINT** (0x0000000fu)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C2_RXPLSEINT** (0x00000010u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C2_TXPLSEINT** (0x00000011u)
- #define **SYSCFG_CFGCHIP1_CAP2SRC_EMAC_C2_MISCINT** (0x00000012u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC** (0x07C00000u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_SHIFT** (0x00000016u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_ECAP1** (0x00000000u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_MCASP0_TXDMA** (0x00000001u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_MCASP0_RXDMA** (0x00000002u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_MCASP1_TXDMA** (0x00000003u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_MCASP1_RXDMA** (0x00000004u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_MCASP2_TXDMA** (0x00000005u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_MCASP2_RXDMA** (0x00000006u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C0_RXTHPLSEINT** (0x00000007u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C0_RXPLSEINT** (0x00000008u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C0_TXPLSEINT** (0x00000009u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C0_MISCINT** (0x0000000au)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C1_RXTHPLSEINT** (0x0000000bu)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C1_RXPLSEINT** (0x0000000cu)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C1_TXPLSEINT** (0x0000000du)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C1_MISCINT** (0x0000000eu)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C2_RXTHPLSEINT** (0x0000000fu)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C2_RXPLSEINT** (0x00000010u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C2_TXPLSEINT** (0x00000011u)
- #define **SYSCFG_CFGCHIP1_CAP1SRC_EMAC_C2_MISCINT** (0x00000012u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC** (0x003E0000u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_SHIFT** (0x00000011u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_ECAP0** (0x00000000u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_MCASP0_TXDMA** (0x00000001u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_MCASP0_RXDMA** (0x00000002u)

- #define **SYSCFG_CFGCHIP1_CAP0SRC_MCASP1_TXDMA** (0x00000003u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_MCASP1_RXDMA** (0x00000004u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_MCASP2_TXDMA** (0x00000005u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_MCASP2_RXDMA** (0x00000006u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C0_RXTHPLSEINT** (0x00000007u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C0_RXPLSEINT** (0x00000008u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C0_TXPLSEINT** (0x00000009u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C0_MISCINT** (0x0000000au)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C1_RXTHPLSEINT** (0x0000000bu)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C1_RXPLSEINT** (0x0000000cu)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C1_TXPLSEINT** (0x0000000du)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C1_MISCINT** (0x0000000eu)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C2_RXTHPLSEINT** (0x0000000fu)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C2_RXPLSEINT** (0x00000010u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C2_TXPLSEINT** (0x00000011u)
- #define **SYSCFG_CFGCHIP1_CAP0SRC_EMAC_C2_MISCINT** (0x00000012u)
- #define **SYSCFG_CFGCHIP1_HPIBYTEAD** (0x00010000u)
- #define **SYSCFG_CFGCHIP1_HPIBYTEAD_SHIFT** (0x00000010u)
- #define **SYSCFG_CFGCHIP1_HPIENA** (0x00008000u)
- #define **SYSCFG_CFGCHIP1_HPIENA_SHIFT** (0x0000000Fu)
- #define **SYSCFG_CFGCHIP1_EDMA31TC0DBS** (0x00006000u)
- #define **SYSCFG_CFGCHIP1_EDMA31TC0DBS_SHIFT** (0x0000000Du)
- #define **SYSCFG_CFGCHIP1_EDMA31TC0DBS_16BYTE** (0x00000000u)
- #define **SYSCFG_CFGCHIP1_EDMA31TC0DBS_32BYTE** (0x00000001u)
- #define **SYSCFG_CFGCHIP1_EDMA31TC0DBS_64BYTE** (0x00000002u)
- #define **SYSCFG_CFGCHIP1_EDMA31TC0DBS_RESERVED** (0x00000003u)
- #define **SYSCFG_CFGCHIP1_TBCLKSYNC** (0x00001000u)
- #define **SYSCFG_CFGCHIP1_TBCLKSYNC_SHIFT** (0x0000000Cu)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0** (0x0000000Fu)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_SHIFT** (0x00000000u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_LOW** (0x00000000u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B0** (0x00000001u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B1** (0x00000002u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B2** (0x00000003u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B3** (0x00000004u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B4** (0x00000005u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B5** (0x00000006u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B6** (0x00000007u)
- #define **SYSCFG_CFGCHIP1_AMUTESEL0_GPIO_B7** (0x00000008u)
- #define **SYSCFG_CFGCHIP2_USB0PHYCLKGD** (0x00020000u)
- #define **SYSCFG_CFGCHIP2_USB0PHYCLKGD_SHIFT** (0x00000011u)
- #define **SYSCFG_CFGCHIP2_USB0VBUSSENSE** (0x00010000u)
- #define **SYSCFG_CFGCHIP2_USB0VBUSSENSE_SHIFT** (0x00000010u)
- #define **SYSCFG_CFGCHIP2_RESET** (0x00008000u)
- #define **SYSCFG_CFGCHIP2_RESET_SHIFT** (0x0000000Fu)
- #define **SYSCFG_CFGCHIP2_USB0OTGMODE** (0x00006000u)
- #define **SYSCFG_CFGCHIP2_USB0OTGMODE_SHIFT** (0x0000000Du)
- #define **SYSCFG_CFGCHIP2_USB0OTGMODE_PHY** (0x00000000u)
- #define **SYSCFG_CFGCHIP2_USB0OTGMODE_USB_HOST** (0x00000001u)
- #define **SYSCFG_CFGCHIP2_USB0OTGMODE_USB_DEVICE** (0x00000002u)
- #define **SYSCFG_CFGCHIP2_USB0OTGMODE_USB_HOST_LOW** (0x00000003u)
- #define **SYSCFG_CFGCHIP2_USB1PHYCLKMUX** (0x00001000u)
- #define **SYSCFG_CFGCHIP2_USB1PHYCLKMUX_SHIFT** (0x0000000Cu)
- #define **SYSCFG_CFGCHIP2_USB0PHYCLKMUX** (0x00000800u)
- #define **SYSCFG_CFGCHIP2_USB0PHYCLKMUX_SHIFT** (0x0000000Bu)

- #define **SYSCFG_CFGCHIP2_USB0PHYPWDN** (0x00000400u)
- #define **SYSCFG_CFGCHIP2_USB0PHYPWDN_SHIFT** (0x0000000Au)
- #define **SYSCFG_CFGCHIP2_USB0OTGPWRDN** (0x00000200u)
- #define **SYSCFG_CFGCHIP2_USB0OTGPWRDN_SHIFT** (0x00000009u)
- #define **SYSCFG_CFGCHIP2_USB0DATPOL** (0x00000100u)
- #define **SYSCFG_CFGCHIP2_USB0DATPOL_SHIFT** (0x00000008u)
- #define **SYSCFG_CFGCHIP2_USB1SUSPENDM** (0x00000080u)
- #define **SYSCFG_CFGCHIP2_USB1SUSPENDM_SHIFT** (0x00000007u)
- #define **SYSCFG_CFGCHIP2_USB0PHY_PLLON** (0x00000040u)
- #define **SYSCFG_CFGCHIP2_USB0PHY_PLLON_SHIFT** (0x00000006u)
- #define **SYSCFG_CFGCHIP2_USB0SESNDEN** (0x00000020u)
- #define **SYSCFG_CFGCHIP2_USB0SESNDEN_SHIFT** (0x00000005u)
- #define **SYSCFG_CFGCHIP2_USB0VBDTCTEN** (0x00000010u)
- #define **SYSCFG_CFGCHIP2_USB0VBDTCTEN_SHIFT** (0x00000004u)
- #define **SYSCFG_CFGCHIP2_USB0REF_FREQ** (0x0000000Fu)
- #define **SYSCFG_CFGCHIP2_USB0REF_FREQ_SHIFT** (0x00000000u)
- #define **SYSCFG_CFGCHIP3_RMII_SEL** (0x00000100u)
- #define **SYSCFG_CFGCHIP3_RMII_SEL_SHIFT** (0x00000008u)
- #define **SYSCFG_CFGCHIP3_UPP_TX_CLKSRC** (0x00000040u)
- #define **SYSCFG_CFGCHIP3_UPP_TX_CLKSRC_SHIFT** (0x00000006u)
- #define **SYSCFG_CFGCHIP3_PLL1_MASTER_LOCK** (0x00000020u)
- #define **SYSCFG_CFGCHIP3_PLL1_MASTER_LOCK_SHIFT** (0x00000005u)
- #define **SYSCFG_CFGCHIP3_ASYNC3_CLKSRC** (0x00000010u)
- #define **SYSCFG_CFGCHIP3_ASYNC3_CLKSRC_SHIFT** (0x00000004u)
- #define **SYSCFG_CFGCHIP3_PRUEVTSEL** (0x00000008u)
- #define **SYSCFG_CFGCHIP3_PRUEVTSEL_SHIFT** (0x00000003u)
- #define **SYSCFG_CFGCHIP3_DIV4P5ENA** (0x00000004u)
- #define **SYSCFG_CFGCHIP3_DIV4P5ENA_SHIFT** (0x00000002u)
- #define **SYSCFG_CFGCHIP3_EMA_CLKSRC** (0x00000002u)
- #define **SYSCFG_CFGCHIP3_EMA_CLKSRC_SHIFT** (0x00000001u)
- #define **SYSCFG_CFGCHIP4_AMUTECLR0** (0x00000001u)
- #define **SYSCFG_CFGCHIP4_AMUTECLR0_SHIFT** (0x00000000u)

### 4.40.1 Detailed Description

SYSCFG0 register definitions for AM1808.

## 4.41 leJOS_EV3/src/ev3/include/hw/hw_syscfg1_AM1808.h File Reference

SYSCFG1 register definitions for AM1808.

### Macros

- #define **SYSCFG1_VTPIO_CTL** (0x0)
- #define **SYSCFG1_DDR_SLEW** (0x4)
- #define **SYSCFG1_DEEPSLEEP** (0x8)
- #define **SYSCFG1_PUPD_ENA** (0xC)
- #define **SYSCFG1_PUPD_SEL** (0x10)
- #define **SYSCFG1_RXACTIVE** (0x14)
- #define **SYSCFG1_PWRDN** (0x18)
- #define **SYSCFG1_VTPIO_CTL_VREFEN** (0x00040000u)
- #define **SYSCFG1_VTPIO_CTL_VREFEN_SHIFT** (0x00000012u)

- #define **SYSCFG1_VTPIO_CTL_VREFTAP** (0x00030000u)
- #define **SYSCFG1_VTPIO_CTL_VREFTAP_SHIFT** (0x00000010u)
- #define **SYSCFG1_VTPIO_CTL_VREFTAP_50_0** (0x00000000u)
- #define **SYSCFG1_VTPIO_CTL_VREFTAP_47_5** (0x00000001u)
- #define **SYSCFG1_VTPIO_CTL_VREFTAP_52_5** (0x00000002u)
- #define **SYSCFG1_VTPIO_CTL_READY** (0x00008000u)
- #define **SYSCFG1_VTPIO_CTL_READY_SHIFT** (0x0000000Fu)
- #define **SYSCFG1_VTPIO_CTL_IOPWRDN** (0x00004000u)
- #define **SYSCFG1_VTPIO_CTL_IOPWRDN_SHIFT** (0x0000000Eu)
- #define **SYSCFG1_VTPIO_CTL_CLKRZ** (0x00002000u)
- #define **SYSCFG1_VTPIO_CTL_CLKRZ_SHIFT** (0x0000000Du)
- #define **SYSCFG1_VTPIO_CTL_FORCEDNP** (0x00001000u)
- #define **SYSCFG1_VTPIO_CTL_FORCEDNP_SHIFT** (0x0000000Cu)
- #define **SYSCFG1_VTPIO_CTL_FORCEDNN** (0x00000800u)
- #define **SYSCFG1_VTPIO_CTL_FORCEDNN_SHIFT** (0x0000000Bu)
- #define **SYSCFG1_VTPIO_CTL_FORCEUPP** (0x00000400u)
- #define **SYSCFG1_VTPIO_CTL_FORCEUPP_SHIFT** (0x0000000Au)
- #define **SYSCFG1_VTPIO_CTL_FORCEUPN** (0x00000200u)
- #define **SYSCFG1_VTPIO_CTL_FORCEUPN_SHIFT** (0x00000009u)
- #define **SYSCFG1_VTPIO_CTL_PWRSAVE** (0x00000100u)
- #define **SYSCFG1_VTPIO_CTL_PWRSAVE_SHIFT** (0x00000008u)
- #define **SYSCFG1_VTPIO_CTL_LOCK** (0x00000080u)
- #define **SYSCFG1_VTPIO_CTL_LOCK_SHIFT** (0x00000007u)
- #define **SYSCFG1_VTPIO_CTL_POWERDN** (0x00000040u)
- #define **SYSCFG1_VTPIO_CTL_POWERDN_SHIFT** (0x00000006u)
- #define **SYSCFG1_VTPIO_CTL_D0** (0x00000020u)
- #define **SYSCFG1_VTPIO_CTL_D0_SHIFT** (0x00000005u)
- #define **SYSCFG1_VTPIO_CTL_D1** (0x00000010u)
- #define **SYSCFG1_VTPIO_CTL_D1_SHIFT** (0x00000004u)
- #define **SYSCFG1_VTPIO_CTL_D2** (0x00000008u)
- #define **SYSCFG1_VTPIO_CTL_D2_SHIFT** (0x00000003u)
- #define **SYSCFG1_VTPIO_CTL_F0** (0x00000004u)
- #define **SYSCFG1_VTPIO_CTL_F0_SHIFT** (0x00000002u)
- #define **SYSCFG1_VTPIO_CTL_F1** (0x00000002u)
- #define **SYSCFG1_VTPIO_CTL_F1_SHIFT** (0x00000001u)
- #define **SYSCFG1_VTPIO_CTL_F2** (0x00000001u)
- #define **SYSCFG1_VTPIO_CTL_F2_SHIFT** (0x00000000u)
- #define **SYSCFG1_DDR_SLEW_ODT_TERMON** (0x00000C00u)
- #define **SYSCFG1_DDR_SLEW_ODT_TERMON_SHIFT** (0x0000000Au)
- #define **SYSCFG1_DDR_SLEW_ODT_TERMOFF** (0x00000300u)
- #define **SYSCFG1_DDR_SLEW_ODT_TERMOFF_SHIFT** (0x00000008u)
- #define **SYSCFG1_DDR_SLEW_DDR_PDENA** (0x00000020u)
- #define **SYSCFG1_DDR_SLEW_DDR_PDENA_SHIFT** (0x00000005u)
- #define **SYSCFG1_DDR_SLEW_CMOSEN** (0x00000010u)
- #define **SYSCFG1_DDR_SLEW_CMOSEN_SHIFT** (0x00000004u)
- #define **SYSCFG1_DDR_SLEW_DDR_DATASLEW** (0x0000000Cu)
- #define **SYSCFG1_DDR_SLEW_DDR_DATASLEW_SHIFT** (0x00000002u)
- #define **SYSCFG1_DDR_SLEW_DDR_CMDSLEW** (0x00000003u)
- #define **SYSCFG1_DDR_SLEW_DDR_CMDSLEW_SHIFT** (0x00000000u)
- #define **SYSCFG1_DEEPSLEEP_SLEEPENABLE** (0x80000000u)
- #define **SYSCFG1_DEEPSLEEP_SLEEPENABLE_SHIFT** (0x0000001Fu)
- #define **SYSCFG1_DEEPSLEEP_SLEEPCOMPLETE** (0x40000000u)
- #define **SYSCFG1_DEEPSLEEP_SLEEPCOMPLETE_SHIFT** (0x0000001Eu)
- #define **SYSCFG1_DEEPSLEEP_SLEEPCOUNT** (0x0000FFFFu)
- #define **SYSCFG1_DEEPSLEEP_SLEEPCOUNT_SHIFT** (0x00000000u)

- #define **SYSCFG1_PUPD_ENA_PUPDENA31** (0x80000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA31_SHIFT** (0x0000001Fu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA30** (0x40000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA30_SHIFT** (0x0000001Eu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA29** (0x20000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA29_SHIFT** (0x0000001Du)
- #define **SYSCFG1_PUPD_ENA_PUPDENA28** (0x10000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA28_SHIFT** (0x0000001Cu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA27** (0x08000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA27_SHIFT** (0x0000001Bu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA26** (0x04000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA26_SHIFT** (0x0000001Au)
- #define **SYSCFG1_PUPD_ENA_PUPDENA25** (0x02000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA25_SHIFT** (0x00000019u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA24** (0x01000000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA24_SHIFT** (0x00000018u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA23** (0x00800000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA23_SHIFT** (0x00000017u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA22** (0x00400000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA22_SHIFT** (0x00000016u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA21** (0x00200000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA21_SHIFT** (0x00000015u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA20** (0x00100000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA20_SHIFT** (0x00000014u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA19** (0x00080000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA19_SHIFT** (0x00000013u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA18** (0x00040000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA18_SHIFT** (0x00000012u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA17** (0x00020000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA17_SHIFT** (0x00000011u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA16** (0x00010000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA16_SHIFT** (0x00000010u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA15** (0x00008000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA15_SHIFT** (0x0000000Fu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA14** (0x00004000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA14_SHIFT** (0x0000000Eu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA13** (0x00002000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA13_SHIFT** (0x0000000Du)
- #define **SYSCFG1_PUPD_ENA_PUPDENA12** (0x00001000u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA12_SHIFT** (0x0000000Cu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA11** (0x00000800u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA11_SHIFT** (0x0000000Bu)
- #define **SYSCFG1_PUPD_ENA_PUPDENA10** (0x00000400u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA10_SHIFT** (0x0000000Au)
- #define **SYSCFG1_PUPD_ENA_PUPDENA9** (0x00000200u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA9_SHIFT** (0x00000009u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA8** (0x00000100u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA8_SHIFT** (0x00000008u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA7** (0x00000080u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA7_SHIFT** (0x00000007u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA6** (0x00000040u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA6_SHIFT** (0x00000006u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA5** (0x00000020u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA5_SHIFT** (0x00000005u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA4** (0x00000010u)

- #define **SYSCFG1_PUPD_ENA_PUPDENA4_SHIFT** (0x00000004u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA3** (0x00000008u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA3_SHIFT** (0x00000003u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA2** (0x00000004u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA2_SHIFT** (0x00000002u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA1** (0x00000002u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA1_SHIFT** (0x00000001u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA0** (0x00000001u)
- #define **SYSCFG1_PUPD_ENA_PUPDENA0_SHIFT** (0x00000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL31** (0x80000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL31_SHIFT** (0x0000001Fu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL30** (0x40000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL30_SHIFT** (0x0000001Eu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL29** (0x20000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL29_SHIFT** (0x0000001Du)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL28** (0x10000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL28_SHIFT** (0x0000001Cu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL27** (0x08000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL27_SHIFT** (0x0000001Bu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL26** (0x04000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL26_SHIFT** (0x0000001Au)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL25** (0x02000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL25_SHIFT** (0x00000019u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL24** (0x01000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL24_SHIFT** (0x00000018u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL23** (0x00800000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL23_SHIFT** (0x00000017u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL22** (0x00400000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL22_SHIFT** (0x00000016u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL21** (0x00200000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL21_SHIFT** (0x00000015u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL20** (0x00100000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL20_SHIFT** (0x00000014u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL19** (0x00080000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL19_SHIFT** (0x00000013u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL18** (0x00040000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL18_SHIFT** (0x00000012u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL17** (0x00020000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL17_SHIFT** (0x00000011u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL16** (0x00010000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL16_SHIFT** (0x00000010u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL15** (0x00008000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL15_SHIFT** (0x0000000Fu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL14** (0x00004000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL14_SHIFT** (0x0000000Eu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL13** (0x00002000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL13_SHIFT** (0x0000000Du)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL12** (0x00001000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL12_SHIFT** (0x0000000Cu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL11** (0x00000800u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL11_SHIFT** (0x0000000Bu)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL10** (0x00000400u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL10_SHIFT** (0x0000000Au)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL9** (0x00000200u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL9_SHIFT** (0x00000009u)

- #define **SYSCFG1_PUPD_SEL_PUPDSEL8** (0x00000100u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL8_SHIFT** (0x00000008u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL7** (0x00000080u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL7_SHIFT** (0x00000007u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL6** (0x00000040u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL6_SHIFT** (0x00000006u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL5** (0x00000020u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL5_SHIFT** (0x00000005u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL4** (0x00000010u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL4_SHIFT** (0x00000004u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL4_PULLDOWN** (0x00000000u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL4_PULLUP** (0x00000001u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL3** (0x00000008u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL3_SHIFT** (0x00000003u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL2** (0x00000004u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL2_SHIFT** (0x00000002u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL1** (0x00000002u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL1_SHIFT** (0x00000001u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL0** (0x00000001u)
- #define **SYSCFG1_PUPD_SEL_PUPDSEL0_SHIFT** (0x00000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE31** (0x80000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE31_SHIFT** (0x0000001Fu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE30** (0x40000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE30_SHIFT** (0x0000001Eu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE29** (0x20000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE29_SHIFT** (0x0000001Du)
- #define **SYSCFG1_RXACTIVE_RXACTIVE28** (0x10000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE28_SHIFT** (0x0000001Cu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE27** (0x08000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE27_SHIFT** (0x0000001Bu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE26** (0x04000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE26_SHIFT** (0x0000001Au)
- #define **SYSCFG1_RXACTIVE_RXACTIVE25** (0x02000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE25_SHIFT** (0x00000019u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE24** (0x01000000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE24_SHIFT** (0x00000018u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE23** (0x00800000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE23_SHIFT** (0x00000017u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE22** (0x00400000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE22_SHIFT** (0x00000016u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE21** (0x00200000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE21_SHIFT** (0x00000015u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE20** (0x00100000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE20_SHIFT** (0x00000014u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE19** (0x00080000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE19_SHIFT** (0x00000013u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE18** (0x00040000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE18_SHIFT** (0x00000012u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE17** (0x00020000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE17_SHIFT** (0x00000011u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE16** (0x00010000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE16_SHIFT** (0x00000010u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE15** (0x00008000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE15_SHIFT** (0x0000000Fu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE14** (0x00004000u)

- #define **SYSCFG1_RXACTIVE_RXACTIVE14_SHIFT** (0x0000000Eu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE13** (0x00002000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE13_SHIFT** (0x0000000Du)
- #define **SYSCFG1_RXACTIVE_RXACTIVE12** (0x00001000u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE12_SHIFT** (0x0000000Cu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE11** (0x00000800u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE11_SHIFT** (0x0000000Bu)
- #define **SYSCFG1_RXACTIVE_RXACTIVE10** (0x00000400u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE10_SHIFT** (0x0000000Au)
- #define **SYSCFG1_RXACTIVE_RXACTIVE9** (0x00000200u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE9_SHIFT** (0x00000009u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE8** (0x00000100u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE8_SHIFT** (0x00000008u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE7** (0x00000080u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE7_SHIFT** (0x00000007u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE6** (0x00000040u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE6_SHIFT** (0x00000006u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE5** (0x00000020u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE5_SHIFT** (0x00000005u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE4** (0x00000010u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE4_SHIFT** (0x00000004u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE3** (0x00000008u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE3_SHIFT** (0x00000003u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE2** (0x00000004u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE2_SHIFT** (0x00000002u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE1** (0x00000002u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE1_SHIFT** (0x00000001u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE0** (0x00000001u)
- #define **SYSCFG1_RXACTIVE_RXACTIVE0_SHIFT** (0x00000000u)
- #define **SYSCFG1_PWRDN_SATACLK_PWRDN** (0x00000001u)
- #define **SYSCFG1_PWRDN_SATACLK_PWRDN_SHIFT** (0x00000000u)

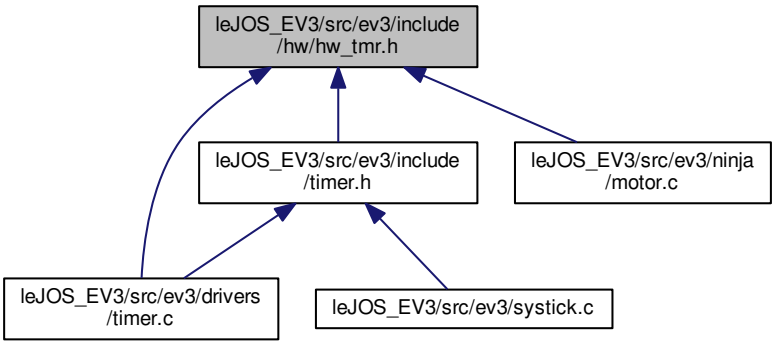
### 4.41.1 Detailed Description

SYSCFG1 register definitions for AM1808.

## 4.42 leJOS_EV3/src/ev3/include/hw/hw_tmr.h File Reference

This file contains the Register Descriptions for Timer.

This graph shows which files directly or indirectly include this file:




## Macros

- #define **TMR_REVID** (0x0)
- #define **TMR_EMUMGT** (0x4)
- #define **TMR_GPINTGPEN** (0x8)
- #define **TMR_GPDATGPDIR** (0xC)
- #define **TMR_TIM12** (0x10)
- #define **TMR_TIM34** (0x14)
- #define **TMR_PRD12** (0x18)
- #define **TMR_PRD34** (0x1C)
- #define **TMR_TCR** (0x20)
- #define **TMR_TGCR** (0x24)
- #define **TMR_WDTCR** (0x28)
- #define **TMR_REL12** (0x34)
- #define **TMR_REL34** (0x38)
- #define **TMR_CAP12** (0x3C)
- #define **TMR_CAP34** (0x40)
- #define **TMR_INTCTLSTAT** (0x44)
- #define **TMR_CMP**(n) (0x60 + (n ∗ 4))
- #define **TMR_REVID_REV** (0xFFFFFFFFu)
- #define **TMR_REVID_REV_SHIFT** (0x00000000u)
- #define **TMR_EMUMGT_SOFT** (0x00000002u)
- #define **TMR_EMUMGT_SOFT_SHIFT** (0x00000001u)
- #define **TMR_EMUMGT_SOFT_IMMEDIATE** (0x00000000u)
- #define **TMR_EMUMGT_SOFT_INCREMENT** (0x00000001u)
- #define **TMR_EMUMGT_FREE** (0x00000001u)
- #define **TMR_EMUMGT_FREE_SHIFT** (0x00000000u)
- #define **TMR_GPINTGPEN_GPENO34** (0x02000000u)
- #define **TMR_GPINTGPEN_GPENO34_SHIFT** (0x00000019u)
- #define **TMR_GPINTGPEN_GPENI34** (0x01000000u)
- #define **TMR_GPINTGPEN_GPENI34_SHIFT** (0x00000018u)
- #define **TMR_GPINTGPEN_GPENO12** (0x00020000u)
- #define **TMR_GPINTGPEN_GPENO12_SHIFT** (0x00000011u)
- #define **TMR_GPINTGPEN_GPENI12** (0x00010000u)
- #define **TMR_GPINTGPEN_GPENI12_SHIFT** (0x00000010u)

- #define **TMR_GPINTGPEN_GPINT34INVO** (0x00002000u)
- #define **TMR_GPINTGPEN_GPINT34INVO_SHIFT** (0x0000000Du)
- #define **TMR_GPINTGPEN_GPINT34INVI** (0x00001000u)
- #define **TMR_GPINTGPEN_GPINT34INVI_SHIFT** (0x0000000Cu)
- #define **TMR_GPINTGPEN_GPINT34ENO** (0x00000200u)
- #define **TMR_GPINTGPEN_GPINT34ENO_SHIFT** (0x00000009u)
- #define **TMR_GPINTGPEN_GPINT34ENI** (0x00000100u)
- #define **TMR_GPINTGPEN_GPINT34ENI_SHIFT** (0x00000008u)
- #define **TMR_GPINTGPEN_GPINT12INVO** (0x00000020u)
- #define **TMR_GPINTGPEN_GPINT12INVO_SHIFT** (0x00000005u)
- #define **TMR_GPINTGPEN_GPINT12INVI** (0x00000010u)
- #define **TMR_GPINTGPEN_GPINT12INVI_SHIFT** (0x00000004u)
- #define **TMR_GPINTGPEN_GPINT12ENO** (0x00000002u)
- #define **TMR_GPINTGPEN_GPINT12ENO_SHIFT** (0x00000001u)
- #define **TMR_GPINTGPEN_GPINT12ENI** (0x00000001u)
- #define **TMR_GPINTGPEN_GPINT12ENI_SHIFT** (0x00000000u)
- #define **TMR_GPDATGPDIR_GPDIRO34** (0x02000000u)
- #define **TMR_GPDATGPDIR_GPDIRO34_SHIFT** (0x00000019u)
- #define **TMR_GPDATGPDIR_GPDIRI34** (0x01000000u)
- #define **TMR_GPDATGPDIR_GPDIRI34_SHIFT** (0x00000018u)
- #define **TMR_GPDATGPDIR_GPDIRO12** (0x00020000u)
- #define **TMR_GPDATGPDIR_GPDIRO12_SHIFT** (0x00000011u)
- #define **TMR_GPDATGPDIR_GPDIRI12** (0x00010000u)
- #define **TMR_GPDATGPDIR_GPDIRI12_SHIFT** (0x00000010u)
- #define **TMR_GPDATGPDIR_GPDATO34** (0x00000200u)
- #define **TMR_GPDATGPDIR_GPDATO34_SHIFT** (0x00000009u)
- #define **TMR_GPDATGPDIR_GPDATI34** (0x00000100u)
- #define **TMR_GPDATGPDIR_GPDATI34_SHIFT** (0x00000008u)
- #define **TMR_GPDATGPDIR_GPDATO12** (0x00000002u)
- #define **TMR_GPDATGPDIR_GPDATO12_SHIFT** (0x00000001u)
- #define **TMR_GPDATGPDIR_GPDATI12** (0x00000001u)
- #define **TMR_GPDATGPDIR_GPDATI12_SHIFT** (0x00000000u)
- #define **TMR_TIM12_TIM12** (0xFFFFFFFFu)
- #define **TMR_TIM12_TIM12_SHIFT** (0x00000000u)
- #define **TMR_TIM34_TIM34** (0xFFFFFFFFu)
- #define **TMR_TIM34_TIM34_SHIFT** (0x00000000u)
- #define **TMR_PRD12_PRD12** (0xFFFFFFFFu)
- #define **TMR_PRD12_PRD12_SHIFT** (0x00000000u)
- #define **TMR_PRD34_PRD34** (0xFFFFFFFFu)
- #define **TMR_PRD34_PRD34_SHIFT** (0x00000000u)
- #define **TMR_TCR_CAPEVTMODE34** (0x30000000u)
- #define **TMR_TCR_CAPEVTMODE34_SHIFT** (0x0000001Cu)
- #define **TMR_TCR_CAPEVTMODE34_RISE** (0x00000000u)
- #define **TMR_TCR_CAPEVTMODE34_FALL** (0x00000001u)
- #define **TMR_TCR_CAPEVTMODE34_BOTH** (0x00000002u)
- #define **TMR_TCR_CAPMODE34** (0x08000000u)
- #define **TMR_TCR_CAPMODE34_SHIFT** (0x0000001Bu)
- #define **TMR_TCR_READRSTMODE34** (0x04000000u)
- #define **TMR_TCR_READRSTMODE34_SHIFT** (0x0000001Au)
- #define **TMR_TCR_TIEN34** (0x02000000u)
- #define **TMR_TCR_TIEN34_SHIFT** (0x00000019u)
- #define **TMR_TCR_CLKSRC34** (0x01000000u)
- #define **TMR_TCR_CLKSRC34_SHIFT** (0x00000018u)
- #define **TMR_TCR_ENAMODE34** (0x00C00000u)
- #define **TMR_TCR_ENAMODE34_SHIFT** (0x00000016u)

- #define **TMR_TCR_ENAMODE34_EN_ONCE** (0x00000001u)
- #define **TMR_TCR_ENAMODE34_EN_CONT** (0x00000002u)
- #define **TMR_TCR_ENAMODE34_EN_CONTRELOAD** (0x00000003u)
- #define **TMR_TCR_PWID34** (0x00300000u)
- #define **TMR_TCR_PWID34_SHIFT** (0x00000014u)
- #define **TMR_TCR_PWID34_ONE_CLK** (0x00000000u)
- #define **TMR_TCR_PWID34_TWO_CLK** (0x00000001u)
- #define **TMR_TCR_PWID34_THREE_CLK** (0x00000002u)
- #define **TMR_TCR_PWID34_FOUR_CLK** (0x00000003u)
- #define **TMR_TCR_CP34** (0x00080000u)
- #define **TMR_TCR_CP34_SHIFT** (0x00000013u)
- #define **TMR_TCR_INVINP34** (0x00040000u)
- #define **TMR_TCR_INVINP34_SHIFT** (0x00000012u)
- #define **TMR_TCR_INVOUTP34** (0x00020000u)
- #define **TMR_TCR_INVOUTP34_SHIFT** (0x00000011u)
- #define **TMR_TCR_TSTAT34** (0x00010000u)
- #define **TMR_TCR_TSTAT34_SHIFT** (0x00000010u)
- #define **TMR_TCR_CAPEVTMODE12** (0x00003000u)
- #define **TMR_TCR_CAPEVTMODE12_SHIFT** (0x0000000Cu)
- #define **TMR_TCR_CAPEVTMODE12_RISE** (0x00000000u)
- #define **TMR_TCR_CAPEVTMODE12_FALL** (0x00000001u)
- #define **TMR_TCR_CAPEVTMODE12_BOTH** (0x00000002u)
- #define **TMR_TCR_CAPMODE12** (0x00000800u)
- #define **TMR_TCR_CAPMODE12_SHIFT** (0x0000000Bu)
- #define **TMR_TCR_READRSTMODE12** (0x00000400u)
- #define **TMR_TCR_READRSTMODE12_SHIFT** (0x0000000Au)
- #define **TMR_TCR_TIEN12** (0x00000200u)
- #define **TMR_TCR_TIEN12_SHIFT** (0x00000009u)
- #define **TMR_TCR_CLKSRC12** (0x00000100u)
- #define **TMR_TCR_CLKSRC12_SHIFT** (0x00000008u)
- #define **TMR_TCR_ENAMODE12** (0x000000C0u)
- #define **TMR_TCR_ENAMODE12_SHIFT** (0x00000006u)
- #define **TMR_TCR_ENAMODE12_EN_ONCE** (0x00000001u)
- #define **TMR_TCR_ENAMODE12_EN_CONT** (0x00000002u)
- #define **TMR_TCR_ENAMODE12_EN_CONTRELOAD** (0x00000003u)
- #define **TMR_TCR_PWID12** (0x00000030u)
- #define **TMR_TCR_PWID12_SHIFT** (0x00000004u)
- #define **TMR_TCR_PWID12_ONE_CLK** (0x00000000u)
- #define **TMR_TCR_PWID12_TWO_CLK** (0x00000001u)
- #define **TMR_TCR_PWID12_THREE_CLK** (0x00000002u)
- #define **TMR_TCR_PWID12_FOUR_CLK** (0x00000003u)
- #define **TMR_TCR_CP12** (0x00000008u)
- #define **TMR_TCR_CP12_SHIFT** (0x00000003u)
- #define **TMR_TCR_INVINP12** (0x00000004u)
- #define **TMR_TCR_INVINP12_SHIFT** (0x00000002u)
- #define **TMR_TCR_INVOUTP12** (0x00000002u)
- #define **TMR_TCR_INVOUTP12_SHIFT** (0x00000001u)
- #define **TMR_TCR_TSTAT12** (0x00000001u)
- #define **TMR_TCR_TSTAT12_SHIFT** (0x00000000u)
- #define **TMR_TGCR_TDDR34** (0x0000F000u)
- #define **TMR_TGCR_TDDR34_SHIFT** (0x0000000Cu)
- #define **TMR_TGCR_PSC34** (0x00000F00u)
- #define **TMR_TGCR_PSC34_SHIFT** (0x00000008u)
- #define **TMR_TGCR_PLUSEN** (0x00000010u)
- #define **TMR_TGCR_PLUSEN_SHIFT** (0x00000004u)

- #define **TMR_TGCR_TIMMODE** (0x0000000Cu)
- #define **TMR_TGCR_TIMMODE_SHIFT** (0x00000002u)
- #define **TMR_TGCR_TIMMODE_64BIT_GPT** (0x00000000u)
- #define **TMR_TGCR_TIMMODE_32BIT_UNCHAIN** (0x00000001u)
- #define **TMR_TGCR_TIMMODE_64BIT_WDT** (0x00000002u)
- #define **TMR_TGCR_TIMMODE_32_CHAIN** (0x00000003u)
- #define **TMR_TGCR_TIM34RS** (0x00000002u)
- #define **TMR_TGCR_TIM34RS_SHIFT** (0x00000001u)
- #define **TMR_TGCR_TIM12RS** (0x00000001u)
- #define **TMR_TGCR_TIM12RS_SHIFT** (0x00000000u)
- #define **TMR_WDTCR_WDKEY** (0xFFFF0000u)
- #define **TMR_WDTCR_WDKEY_SHIFT** (0x00000010u)
- #define **TMR_WDTCR_WDKEY_CMD1** (0x0000A5C6u)
- #define **TMR_WDTCR_WDKEY_CMD2** (0x0000DA7Eu)
- #define **TMR_WDTCR_WDFLAG** (0x00008000u)
- #define **TMR_WDTCR_WDFLAG_SHIFT** (0x0000000Fu)
- #define **TMR_WDTCR_WDEN** (0x00004000u)
- #define **TMR_WDTCR_WDEN_SHIFT** (0x0000000Eu)
- #define **TMR_REL12_REL12** (0xFFFFFFFFu)
- #define **TMR_REL12_REL12_SHIFT** (0x00000000u)
- #define **TMR_REL34_REL34** (0xFFFFFFFFu)
- #define **TMR_REL34_REL34_SHIFT** (0x00000000u)
- #define **TMR_CAP12_CAP12** (0xFFFFFFFFu)
- #define **TMR_CAP12_CAP12_SHIFT** (0x00000000u)
- #define **TMR_CAP34_CAP34** (0xFFFFFFFFu)
- #define **TMR_CAP34_CAP34_SHIFT** (0x00000000u)
- #define **TMR_INTCTLSTAT_EVTINTSTAT34** (0x00080000u)
- #define **TMR_INTCTLSTAT_EVTINTSTAT34_SHIFT** (0x00000013u)
- #define **TMR_INTCTLSTAT_EVTINTEN34** (0x00040000u)
- #define **TMR_INTCTLSTAT_EVTINTEN34_SHIFT** (0x00000012u)
- #define **TMR_INTCTLSTAT_PRDINTSTAT34** (0x00020000u)
- #define **TMR_INTCTLSTAT_PRDINTSTAT34_SHIFT** (0x00000011u)
- #define **TMR_INTCTLSTAT_PRDINTEN34** (0x00010000u)
- #define **TMR_INTCTLSTAT_PRDINTEN34_SHIFT** (0x00000010u)
- #define **TMR_INTCTLSTAT_EVTINTSTAT12** (0x00000008u)
- #define **TMR_INTCTLSTAT_EVTINTSTAT12_SHIFT** (0x00000003u)
- #define **TMR_INTCTLSTAT_EVTINTEN12** (0x00000004u)
- #define **TMR_INTCTLSTAT_EVTINTEN12_SHIFT** (0x00000002u)
- #define **TMR_INTCTLSTAT_PRDINTSTAT12** (0x00000002u)
- #define **TMR_INTCTLSTAT_PRDINTSTAT12_SHIFT** (0x00000001u)
- #define **TMR_INTCTLSTAT_PRDINTEN12** (0x00000001u)
- #define **TMR_INTCTLSTAT_PRDINTEN12_SHIFT** (0x00000000u)
- #define **TMR_CMP0_CMP** (0xFFFFFFFFu)
- #define **TMR_CMP0_CMP_SHIFT** (0x00000000u)

### 4.42.1 Detailed Description

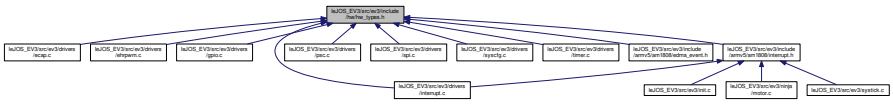
This file contains the Register Descriptions for Timer.

==========

## 4.43 leJOS_EV3/src/ev3/include/hw/hw_types.h File Reference

Common type definitions and macros.

This graph shows which files directly or indirectly include this file:



### Macros

- #define **true** 1
- #define **false** 0
- #define **NULL** ((void∗) 0)
- #define **HWREG**(x) (∗((volatile unsigned int ∗)(x)))
- #define **HWREGH**(x) (∗((volatile unsigned short ∗)(x)))
- #define **HWREGB**(x) (∗((volatile unsigned char ∗)(x)))
- #define **HWREGBITW**(x, b)
- #define **HWREGBITH**(x, b)
- #define **HWREGBITB**(x, b)
- #define **TRUE** 1
- #define **FALSE** 0

### Typedefs

- typedef unsigned char **tBoolean**

### 4.43.1 Detailed Description

Common type definitions and macros.

### 4.43.2 Macro Definition Documentation

#### 4.43.2.1 #define HWREGBITB( *x, b* )

**Value:**

```
HWREGB(((unsigned int)(x) & 0xF0000000) | 0x02000000 |                \
               (((unsigned int)(x) & 0x000FFFFF) << 5) | ((b) << 2))
```

#### 4.43.2.2 #define HWREGBITH( *x, b* )

**Value:**

```
HWREGH(((unsigned int)(x) & 0xF0000000) | 0x02000000 |                \
               (((unsigned int)(x) & 0x000FFFFF) << 5) | ((b) << 2))
```

#### 4.43.2.3 #define HWREGBITW( *x*, *b* )

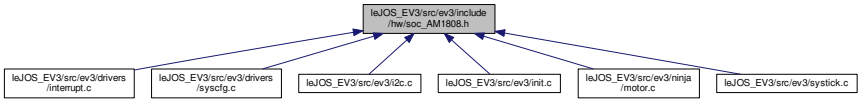
**Value:**

```
HWREG(((unsigned int)(x) & 0xF0000000) | 0x02000000 |                \
              (((unsigned int)(x) & 0x000FFFFF) << 5) | ((b) << 2))
```

## 4.44 leJOS_EV3/src/ev3/include/hw/soc_AM1808.h File Reference

This file contains the peripheral information for AM1808 SOC.

This graph shows which files directly or indirectly include this file:




### Macros

- #define SOC_UPP_PER_CNT 1

  *Number of UPP instances.*
- #define SOC_HPI_PER_CNT 1

  *Number of UHPI instances.*
- #define SOC_MCASP_PER_CNT 1

  *Number of McASP instances.*
- #define SOC_TMR_PER_CNT 4

  *Number of TIMER instances.*
- #define SOC_PSC_PER_CNT 2

  *Number of PSC instances.*
- #define SOC_UART_PER_CNT 3

  *Number of UART instances.*
- #define SOC_SPI_PER_CNT 2

  *Number of SPI instances.*
- #define SOC_I2C_PER_CNT 2

  *Number of I2C instances.*
- #define SOC_PLLC_PER_CNT 2

  *Number of PLL instances.*
- #define SOC_MMCSD_PER_CNT 2

  *Number of MMCSD instances.*
- #define SOC_LCDC_PER_CNT 1

  *Number of LCDC instances.*
- #define SOC_MCBSP_PER_CNT 2

  *Number of Mcbsp instances.*
- #define SOC_EDMA3CC_CNT 2

  *Number of EDMA3 CC instances.*
- #define SOC_EDMA3TC_CNT 3

  *Number of EDMA3 TC instances.*
- #define SOC_EMIFA_PER_CNT 1

  *Number of EMIFA instances.*

- #define SOC_EMIFB_PER_CNT 1

  *Number of EMIFB instances.*
- #define SOC_EMAC_PER_CNT 1

  *Number of EMAC instances.*
- #define SOC_MDIO_PER_CNT 1

  *Number of MDIO instances.*
- #define SOC_EHRPWM_PER_CNT 2

  *Number of EHRPWM instances.*
- #define SOC_ECAP_PER_CNT 3

  *Number of ECAP instances.*
- #define SOC_CPGMACSSR_PER_CNT 1

  *Number of CPGMAC instances.*
- #define SOC_CPPI_PER_CNT 1

  *Number of CPPI instances.*
- #define SOC_USB_PER_CNT 2

  *Number of USB instances.*
- #define SOC_VPIF_PER_CNT 1

  *Number of VPIF instances.*
- #define SOC_INTC_PER_CNT 1

  *Number of INTC instances.*
- #define SOC_AINTC_PER_CNT 1

  *Number of AINTC instances.*
- #define SOC_SATA_PER_CNT 1

  *Number of SATA instances.*
- #define SOC_RTC_PER_CNT 1

  *Number of RTC instances.*
- #define SOC_GPIO_PER_CNT 1

  *Number of GPIO instances.*
- #define SOC_SYSCFG_PER_CNT 2

  *Number of SYSCFG instances.*
- #define SOC_HPI (0)

  *Peripheral Instances of UHPI instances.*
- #define SOC_MCASP_0 (0)

  *Peripheral Instances of McASP instances.*
- #define SOC_EDMA3CC_0 (0)

  *Peripheral Instance of EDMA CC instances.*
- #define **SOC_EDMA3CC_1** (1)
- #define SOC_EDMA3TC_0 (0)

  *Peripheral Instance of EDMA TC instances.*
- #define **SOC_EDMA3TC_1** (1)
- #define SOC_TMR_0 (0)

  *Peripheral Instance of Timer 64 instances.*
- #define **SOC_TMR_1** (1)
- #define **SOC_TMR_2** (2)
- #define **SOC_TMR_3** (3)
- #define SOC_PSC_0 (0)

  *Peripheral Instances of PSC instances.*
- #define **SOC_PSC_1** (1)
- #define SOC_UART_0 (0)

  *Peripheral Instances of UART instances.*
- #define **SOC_UART_1** (1)

- #define **SOC_UART_2** (2)
- #define SOC_SPI_0 (0)

  *Peripheral Instances of SPI instances.*
- #define **SOC_SPI_1** (1)
- #define SOC_I2C_0 (0)

  *Peripheral Instances of I2C instances.*
- #define **SOC_I2C_1** (1)
- #define SOC_MMCSD_0 (0)

  *Peripheral Instances of MMCSD instances.*
- #define **SOC_MMCSD_1** (1)
- #define SOC_LCDC (0)

  *Peripheral Instances of LCDC instances.*
- #define SOC_PLLC_0 (0)

  *Instance number of PLL controller.*
- #define **SOC_PLLC_1** (1)
- #define SOC_EMIFA (0)

  *Peripheral Instance of EMIFA instances.*
- #define SOC_EMAC (0)

  *Peripheral Instance of EMAC instances.*
- #define SOC_MDIO (0)

  *Peripheral Instance of MDIO instances.*
- #define SOC_EHRPWM_0 (0)

  *Peripheral Instance of EHRPWM instances.*
- #define **SOC_EHRPWM_1** (1)
- #define SOC_ECAP_0 (0)

  *Peripheral Instance of ECAP instances.*
- #define **SOC_ECAP_1** (1)
- #define **SOC_ECAP_2** (2)
- #define SOC_USB_0 (0)

  *Peripheral Instance of USB instances.*
- #define **SOC_USB_1** (1)
- #define SOC_PRUCORE_0 (0)

  *Peripheral Instance of PRU CORE instances.*
- #define **SOC_PRUCORE_1** (1)
- #define SOC_PRUINTC (0)

  *Peripheral Instance of PRUINTC instances.*
- #define SOC_VPIF (0)

  *Peripheral Instances of VPIF instances.*
- #define SOC_INTC (0)

  *Peripheral Instance of INTC instances.*
- #define SOC_AINTC (0)

  *Peripheral Instance of AINTC instances.*
- #define SOC_RTC (0)

  *Peripheral Instance of RTC instances.*
- #define SOC_GPIO (0)

  *Peripheral Instance of GPIO instances.*
- #define SOC_GPIO_NUM_PINS (144)

  *GPIO pin and bank information.*
- #define **SOC_GPIO_NUM_BANKS** ((SOC_GPIO_NUM_PINS + 15)/16)
- #define SOC_ECTL (0)

  *Peripheral Instance of ECTL instances.*

- #define SOC_SYSCFG (2)

  *Peripheral Instance of SYSCFG instances.*
- #define SOC_INTC_0_REGS (0x01800000)

  *Base address of INTC memory mapped registers.*
- #define SOC_PWRDWN_PDC_REGS (0x01810000)

  *Base address of PDC memory mapped registers.*
- #define SOC_SYS_0_SECURITY_ID_REGS (0x01811000)

  *Base address of SYS - Security ID register.*
- #define SOC_SYS_0_REV_ID_REGS (0x01812000)

  *Base address of SYS - Revision ID register.*
- #define SOC_IDMA_0_REGS (0x01820000)
- #define SOC_EMC_0_REGS (0x01820000)
- #define SOC_CACHE_0_REGS (0x01840000)
- #define SOC_EDMA30CC_0_REGS (0x01C00000)

  *Base address of Channel controller memory mapped registers.*
- #define SOC_EDMA30TC_0_REGS (0x01C08000)

  *Base address of Transfer controller memory mapped registers.*
- #define **SOC_EDMA30TC_1_REGS** (0x01C08400)
- #define SOC_PSC_0_REGS (0x01C10000)

  *Base address of PSC memory mapped registers.*
- #define SOC_PLLC_0_REGS (0x01C11000)

  *PLL controller instance o module address.*
- #define SOC_SYSCFG_0_REGS (0x01C14000)

  *Base address of DEV memory mapped registers.*
- #define SOC_TMR_0_REGS (0x01C20000)

  *Base address of TIMER memory mapped registers.*
- #define **SOC_TMR_1_REGS** (0x01C21000)
- #define SOC_I2C_0_REGS (0x01C22000)

  *Base address of I2C memory mapped registers.*
- #define SOC_RTC_0_REGS (0x01C23000)

  *Base address of RTC memory.*
- #define SOC_MMCSD_0_REGS (0x01C40000)

  *Base address of MMCSD memory mapped registers.*
- #define SOC_SPI_0_REGS (0x01C41000)

  *Base address of SPI memory mapped registers.*
- #define SOC_UART_0_REGS (0x01C42000)

  *Base address of UART memory mapped registers.*
- #define SOC_MCASP_0_CTRL_REGS (0x01D00000)

  *Base address of McASP memory mapped registers.*
- #define **SOC_MCASP_0_FIFO_REGS** (0x01D01000)
- #define **SOC_MCASP_0_DATA_REGS** (0x01D02000)
- #define SOC_UART_1_REGS (0x01D0C000)

  *Base address of UART memory mapped registers.*
- #define **SOC_UART_2_REGS** (0x01D0D000)
- #define SOC_MCBSP_0_CTRL_REGS (0x01D10000)

  *Base address of McBSP memory mapped registers.*
- #define **SOC_MCBSP_0_FIFO_REGS** (0x01D10800)
- #define **SOC_MCBSP_0_DATA_REGS** (0x01F10000)
- #define SOC_MCBSP_1_CTRL_REGS (0x01D11000)

  *Base address of McASP memory mapped registers.*
- #define **SOC_MCBSP_1_FIFO_REGS** (0x01D11800)

- #define **SOC_MCBSP_1_DATA_REGS** (0x01F11000)
- #define **SOC_MPU_0_REGS** (0x01E14000)
- #define **SOC_MPU_1_REGS** (0x01E15000)
- #define SOC_USB_0_REGS (0x01E00000)

  *Base address of USB memory.*
- #define **SOC_USB_1_REGS** (0x01E25000)
- #define SOC_HPI_0_REGS (0x01E10000)

  *Base address of HPI memory mapped registers.*
- #define SOC_LCDC_0_REGS (0x01E13000)

  *Base address of LCDC memory mapped registers.*
- #define SOC_UPP_0_REGS (0x01E16000)

  *Base address of UPP memory mapped registers.*
- #define SOC_VPIF_0_REGS (0x01E17000)

  *Base address of VPIF memory mapped registers.*
- #define SOC_SATA_0_REGS (0x01E18000)

  *Base address of SATA memory mapped registers.*
- #define SOC_PLLC_1_REGS (0X01E1A000)

  *PLL controller instance 1 module address.*
- #define SOC_MMCSD_1_REGS (0x01E1B000)

  *Base address of MMCSD memory mapped registers.*
- #define SOC_EMAC_DSC_CTRL_MOD_RAM (0x01E20000)

  *Base address of EMAC memory.*
- #define **SOC_EMAC_DSC_CTRL_MOD_REG** (0x01E22000)
- #define **SOC_EMAC_DSC_CONTROL_REG** (0x01E23000)
- #define **SOC_MDIO_0_REGS** (0x01E24000)
- #define SOC_PRUCORE_0_REGS (0x01C37000)

  *Base address of PRU Core Regsiters.*
- #define **SOC_PRUCORE_1_REGS** (0x01C37800)
- #define SOC_PRUINTC_0_REGS (0x01C34000)

  *Base address of PRU Interrupt Controller Registers.*
- #define SOC_USB_0_BASE (0x01E00400)

  *Base address of MUSB memmory mapped Registers.*
- #define SOC_USB_0_OTG_BASE (0x01E00000)

  *Base address of OTG memmory mapped Registers.*
- #define SOC_USB_0_PHY_REGS (0x01C14184)
- #define SOC_GPIO_0_REGS (0x01E26000)

  *Base address of GPIO memory mapped registers.*
- #define SOC_PSC_1_REGS (0x01E27000)

  *Base address of PSC memory mapped registers.*
- #define SOC_I2C_1_REGS (0x01E28000)

  *Base address of I2C memory mapped registers.*
- #define SOC_SYSCFG_1_REGS (0x01E2C000)

  *Base address of syscfg registers.*
- #define SOC_EDMA31CC_0_REGS (0x01E30000)

  *Base address of Channel controller memory mapped registers.*
- #define SOC_EDMA31TC_0_REGS (0x01E38000)

  *Base address of Transfer controller memory mapped registers.*
- #define SOC_EHRPWM_0_REGS (0x01F00000)

  *Base address of EPWM memory mapped registers.*
- #define **SOC_EHRPWM_1_REGS** (0x01F02000)
- #define SOC_HRPWM_0_REGS (0x01F01000)

*Base address of EPWM memory mapped registers.*

- #define **SOC_HRPWM_1_REGS** (0x01F03000)
- #define SOC_ECAP_0_REGS (0x01F06000)

*Base address of ECAP memory mapped registers.*

- #define **SOC_ECAP_1_REGS** (0x01F07000)
- #define **SOC_ECAP_2_REGS** (0x01F08000)
- #define SOC_TMR_2_REGS (0x01F0C000)

*Base address of TIMER memory mapped registers.*

- #define **SOC_TMR_3_REGS** (0x01F0D000)
- #define SOC_SPI_1_REGS (0x01F0E000)

*Base address of SPI memory mapped registers.*

- #define SOC_EMIFA_0_REGS (0x68000000)

*Base address of EMIFA memory mapped registers.*

- #define SOC_EMIFA_CS0_ADDR (0x40000000)

*Base address of EMIFA_CS0 memory.*

- #define SOC_EMIFA_CS2_ADDR (0x60000000)

*Base address of EMIFA_CS2 memory.*

- #define SOC_EMIFA_CS3_ADDR (0x62000000)

*Base address of EMIFA_CS3 memory.*

- #define SOC_EMIFA_CS4_ADDR (0x64000000)

*Base address of EMIFA_CS4 memory.*

- #define SOC_EMIFA_CS5_ADDR (0x66000000)

*Base address of EMIFA_CS5 memory.*

- #define SOC_DDR2_0_CTRL_REGS (0xB0000000)

*Base address of DDR memory mapped registers.*

- #define **SOC_DDR2_0_DATA_REGS** (0xC0000000)
- #define SOC_AINTC_0_REGS (0xFFFEE000)

*Base address of AINTC memory mapped registers.*

- #define SOC_MEMPROT_L2_REGS (0x00800000)

*Base address of UMC Memory protection registers.*

- #define SOC_MEMPROT_L1P_REGS (0x00E00000)

*Base address of PMC memory Protection registers.*

- #define SOC_MEMPROT_L1D_REGS (0x00F00000)

*Base address of DMC memory protection registers.*

- #define **SOC_EDMA3_CHA_CNT**
- #define **SOC_EDMA3_QCHA_BASE** SOC_EDMA3_NUM_DMACH /∗ QDMA Channel Base ∗/
- #define **SOC_EDMA3_QCHA_0** (SOC_EDMA3_QCHA_BASE + 0) /∗ QDMA Channel 0 ∗/
- #define **SOC_EDMA3_QCHA_1** (SOC_EDMA3_QCHA_BASE + 1) /∗ QDMA Channel 1 ∗/
- #define **SOC_EDMA3_QCHA_2** (SOC_EDMA3_QCHA_BASE + 2) /∗ QDMA Channel 2 ∗/
- #define **SOC_EDMA3_QCHA_3** (SOC_EDMA3_QCHA_BASE + 3) /∗ QDMA Channel 3 ∗/
- #define **SOC_EDMA3_QCHA_4** (SOC_EDMA3_QCHA_BASE + 4) /∗ QDMA Channel 4 ∗/
- #define **SOC_EDMA3_QCHA_5** (SOC_EDMA3_QCHA_BASE + 5) /∗ QDMA Channel 5 ∗/
- #define **SOC_EDMA3_QCHA_6** (SOC_EDMA3_QCHA_BASE + 6) /∗ QDMA Channel 6 ∗/
- #define **SOC_EDMA3_QCHA_7** (SOC_EDMA3_QCHA_BASE + 7) /∗ QDMA Channel 7 ∗/
- #define **SOC_EDMACC_ANY** -1 /∗ Any instance of EDMACC module∗/
- #define **SOC_EDMACC_0** 0 /∗ EDMACC Instance 0 ∗/
- #define **SOC_EDMA3_QUE_0** 0 /∗ Queue 0 ∗/
- #define **SOC_EDMA3_QUE_1** 1 /∗ Queue 1 ∗/
- #define **SOC_EDMATC_ANY** -1 /∗ Any instance of EDMATC ∗/
- #define **SOC_EDMATC_0** 0 /∗ EDMATC Instance 0 ∗/
- #define **SOC_EDMATC_1** 1 /∗ EDMATC Instance 1 ∗/
- #define **SOC_EDMA3_REGION_GLOBAL** (-1)

- #define **SOC_EDMA3_REGION_0** 0
- #define **SOC_EDMA3_REGION_1** 1
- #define **SOC_EDMA3_REGION_2** 2
- #define **SOC_EDMA3_REGION_3** 3
- #define SOC_DAT_QCHA_0 0

  *Number of Generic Channel instances.*
- #define SOC_DAT_QCHA_1 1
- #define SOC_DAT_QCHA_2 2
- #define SOC_DAT_QCHA_3 3
- #define SOC_DAT_QCHA_4 4
- #define SOC_DAT_QCHA_5 5
- #define SOC_DAT_QCHA_6 6
- #define SOC_DAT_QCHA_7 7
- #define SOC_DAT_PRI_DEFAULT 0 /∗ Queue 0 is default ∗/

  *Enumerations for EDMA Event Queues.*
- #define **SOC_DAT_PRI_0** 0 /∗ Queue 0 ∗/
- #define **SOC_DAT_PRI_1** 1 /∗ Queue 1 ∗/
- #define SOC_DAT_REGION_GLOBAL (-1) /∗ Global Region ∗/

  *Enumeration for EDMA Regions.*
- #define **SOC_DAT_REGION_0** 0 /∗ EDMA Region 0 ∗/
- #define **SOC_DAT_REGION_1** 1 /∗ EDMA Region 1 ∗/
- #define **SOC_DAT_REGION_2** 2 /∗ EDMA Region 2 ∗/
- #define **SOC_DAT_REGION_3** 3 /∗ EDMA Region 3 ∗/
- #define SOC_SYSCLK_1_FREQ (300000000)

  *Enumeration for peripheral frequencies.*
- #define **SOC_SYSCLK_2_FREQ** (SOC_SYSCLK_1_FREQ/2)
- #define **SOC_SYSCLK_3_FREQ** (SOC_SYSCLK_1_FREQ/3)
- #define **SOC_SYSCLK_4_FREQ** (SOC_SYSCLK_1_FREQ/4)
- #define **SOC_ASYNC_2_FREQ** (24000000)
- #define SOC_I2C_0_MODULE_FREQ (SOC_ASYNC_2_FREQ)
- #define **SOC_I2C_1_MODULE_FREQ** (SOC_SYSCLK_4_FREQ)
- #define SOC_MCBSP_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)
- #define **SOC_MCBSP_1_MODULE_FREQ** (SOC_SYSCLK_2_FREQ)
- #define SOC_LCDC_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)
- #define SOC_SPI_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)
- #define **SOC_SPI_1_MODULE_FREQ** (SOC_SYSCLK_2_FREQ)
- #define SOC_UART_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)
- #define **SOC_UART_1_MODULE_FREQ** (SOC_SYSCLK_2_FREQ)
- #define **SOC_UART_2_MODULE_FREQ** (SOC_SYSCLK_2_FREQ)
- #define SOC_EHRPWM_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)
- #define **SOC_EHRPWM_1_MODULE_FREQ** (SOC_SYSCLK_2_FREQ)

### 4.44.1 Detailed Description

This file contains the peripheral information for AM1808 SOC.

==========

### 4.44.2 Macro Definition Documentation

#### 4.44.2.1 #define SOC_CACHE_0_REGS (0x01840000)

#brief Cache Module memory mapped address

#### 4.44.2.2 #define SOC_DAT_PRI_DEFAULT 0 /* Queue 0 is default */

Enumerations for EDMA Event Queues.

There are two Event Queues. Q0 is the highest priority and Q1 is the least priority

#### 4.44.2.3 #define SOC_DAT_QCHA_0 0

Number of Generic Channel instances.

Enumerations for EDMA channels

There are 8 QDMA channels -QDMA Channel 0

#### 4.44.2.4 #define SOC_DAT_QCHA_1 1

QDMA Channel 1

#### 4.44.2.5 #define SOC_DAT_QCHA_2 2

QDMA Channel 2

#### 4.44.2.6 #define SOC_DAT_QCHA_3 3

QDMA Channel 3

#### 4.44.2.7 #define SOC_DAT_QCHA_4 4

QDMA Channel 4

#### 4.44.2.8 #define SOC_DAT_QCHA_5 5

QDMA Channel 5

#### 4.44.2.9 #define SOC_DAT_QCHA_6 6

QDMA Channel 6

#### 4.44.2.10 #define SOC_DAT_QCHA_7 7

QDMA Channel 7

#### 4.44.2.11 #define SOC_DAT_REGION_GLOBAL (-1) /* Global Region */

Enumeration for EDMA Regions.

#### 4.44.2.12 #define SOC_EDMA3_CHA_CNT

**Value:**

```
(SOC_EDMA3_NUM_DMACH + \
                                        SOC_EDMA3_NUM_QDMACH)
```

#### 4.44.2.13 #define SOC_EHRPWM_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)

EHRPWM

#### 4.44.2.14 #define SOC_EMC_0_REGS (0x01820000)

#brief EMC Module memory mapped address

#### 4.44.2.15 #define SOC_I2C_0_MODULE_FREQ (SOC_ASYNC_2_FREQ)

I2C

#### 4.44.2.16 #define SOC_IDMA_0_REGS (0x01820000)

#brief IDMA Module memory mapped address

#### 4.44.2.17 #define SOC_LCDC_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)

LCDC

#### 4.44.2.18 #define SOC_MCBSP_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)

MCBSP

#### 4.44.2.19 #define SOC_SPI_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)

SPI

#### 4.44.2.20 #define SOC_SYSCLK_1_FREQ (300000000)

Enumeration for peripheral frequencies.

#### 4.44.2.21 #define SOC_UART_0_MODULE_FREQ (SOC_SYSCLK_2_FREQ)

UART

#### 4.44.2.22 #define SOC_USB_0_PHY_REGS (0x01C14184)

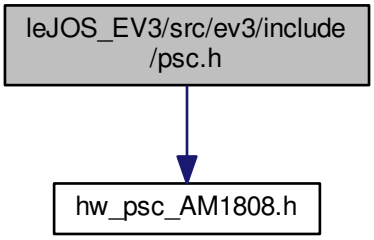
∗brief USB 0 Phy regsister( CFGCHIP2 register) address

## 4.45 leJOS_EV3/src/ev3/include/psc.h File Reference

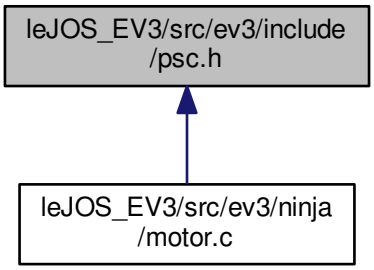
This file contains the function prototypes for the device abstraction layer for PSC. It also contains some related macro definitions and some files to be included.

```
#include "hw_psc_AM1808.h"
```

Include dependency graph for psc.h:




This graph shows which files directly or indirectly include this file:




## Functions

- int PSCModuleControl (unsigned int baseAdd, unsigned int moduleId, unsigned int powerDomain, unsigned int flags)

  *This function sets the requested module in the required state.*
- void **USBModuleClkEnable** (unsigned int ulIndex, unsigned int ulBase)
- void **USBModuleClkDisable** (unsigned int ulIndex, unsigned int ulBase)

### 4.45.1 Detailed Description

This file contains the function prototypes for the device abstraction layer for PSC. It also contains some related macro definitions and some files to be included.

### 4.45.2 Function Documentation

#### 4.45.2.1 int PSCModuleControl ( unsigned int *baseAdd,* unsigned int *moduleId,* unsigned int *powerDomain,* unsigned int *flags* )

This function sets the requested module in the required state.

**Parameters**

| | |
|---|---|
| *baseAdd* | Memory address of the PSC instance used. |
| *moduleId* | The module number of the module to be commanded. |
| *powerDomain* | The power domain of the module to be commanded. |
| *flags* | This contains the flags that is a logical OR of the commands that can be given to a module. |

**Returns**

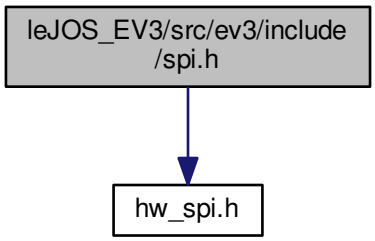
0 in case of successful transition, -1 otherwise.

## 4.46 leJOS_EV3/src/ev3/include/spi.h File Reference

SPI APIs and macros.

```
#include "hw_spi.h"
```

Include dependency graph for spi.h:




### Macros

- #define **SPI_MASTER_MODE** (SPI_SPIGCR1_MASTER | SPI_SPIGCR1_CLKMOD)
- #define **SPI_SLAVE_MODE** 0
- #define **SPI_DATA_FORMAT0** SPI_SPIDAT1_DFSEL_FORMAT0
- #define **SPI_DATA_FORMAT1** SPI_SPIDAT1_DFSEL_FORMAT1
- #define **SPI_DATA_FORMAT2** SPI_SPIDAT1_DFSEL_FORMAT2
- #define **SPI_DATA_FORMAT3** SPI_SPIDAT1_DFSEL_FORMAT3
- #define **SPI_CSHOLD** SPI_SPIDAT1_CSHOLD
- #define **SPI_DELCOUNT_ENABLE** SPI_SPIDAT1_WDEL
- #define **SPI_DELCOUNT_DISABLE** 0
- #define **SPI_CLK_POL_HIGH** SPI_SPIFMT_POLARITY
- #define **SPI_CLK_POL_LOW** 0
- #define **SPI_CLK_INPHASE** 0
- #define **SPI_CLK_OUTOFPHASE** SPI_SPIFMT_PHASE
- #define **SPI_ODD_PARITY** SPI_SPIFMT_PARPOL
- #define **SPI_EVEN_PARITY** 0
- #define **SPI_DATALEN_ERR_INTLVL** SPI_SPILVL_DLENERRLVL
- #define **SPI_TIMEOUT_INTLVL** SPI_SPILVL_TIMEOUTLVL
- #define **SPI_PARITY_ERR_INTLVL** SPI_SPILVL_PARERRLVL
- #define **SPI_DESYNC_SLAVE_INTLVL** SPI_SPILVL_DESYNCLVL

- #define **SPI_BIT_ERR_INTLVL** SPI_SPILVL_BITERRLVL
- #define **SPI_RECV_OVERRUN_INTLVL** SPI_SPILVL_OVRNINTLVL
- #define **SPI_RECV_INTLVL** SPI_SPILVL_RXINTLVL
- #define **SPI_TRANSMIT_INTLVL** SPI_SPILVL_TXINTLVL
- #define **SPI_DATALEN_ERR_INT** SPI_SPIINT0_DLENERRENA
- #define **SPI_TIMEOUT_INT** SPI_SPIINT0_TIMEOUTENA
- #define **SPI_PARITY_ERR_INT** SPI_SPIINT0_PARERRENA
- #define **SPI_DESYNC_SLAVE_INT** SPI_SPIINT0_DESYNCENA
- #define **SPI_BIT_ERR_INT** SPI_SPIINT0_BITERRENA
- #define **SPI_RECV_OVERRUN_INT** SPI_SPIINT0_OVRNINTENA
- #define **SPI_RECV_INT** SPI_SPIINT0_RXINTENA
- #define **SPI_TRANSMIT_INT** SPI_SPIINT0_TXINTENA
- #define **SPI_DMA_REQUEST_ENA_INT** SPI_SPIINT0_DMAREQEN
- #define **SPI_TX_BUF_EMPTY** 0x14
- #define **SPI_RECV_FULL** 0x12
- #define **SPI_ERR** 0x11

## Functions

- void SPIClkConfigure (unsigned int baseAdd, unsigned int moduleClk, unsigned int spiClk, unsigned int dataFormat)
  - *It will configure the prescalar to generate required spi clock*
  .
- void SPIEnable (unsigned int baseAdd)
  - *It will Enable SPI module*
  .
- void SPIReset (unsigned int baseAdd)
  - *It will put SPI in to reset state.*
- void SPIOutOfReset (unsigned int baseAdd)
  - *Brings SPI out of reset state.*
- void SPIModeConfigure (unsigned int baseAdd, unsigned int flag)
  - *Configures SPI to master or slave mode.*
- void SPIPinControl (unsigned int baseAdd, unsigned int idx, unsigned int flag, unsigned int *val)
  - *Configures SPI Pin Control Registers.*
- void SPIDelayConfigure (unsigned int baseAdd, unsigned int c2edelay, unsigned int t2edelay, unsigned int t2cdelay, unsigned int c2tdelay)
  - *Configures SPI CS and ENA Delay in SPIDELAY Register.*
- void SPIDefaultCSSet (unsigned int baseAdd, unsigned char dcsval)
  - *Sets the default value for CS pin(line) when no transmission is is performed by writing to SPIDEF Reg.*
- void SPIDat1Config (unsigned int baseAdd, unsigned int flag, unsigned char cs)
  - *Configures the SPIDat1 Register.It won't write to TX part of the SPIDat1 Register.*
- void SPIConfigClkFormat (unsigned int baseAdd, unsigned int flag, unsigned int dataFormat)
  - *It Configures SPI Clock's Phase and Polarity.*
- void SPITransmitData1 (unsigned int baseAdd, unsigned int data)
  - *Trasmits Data from TX part of SPIDAT1 register.*
- void SPICSTimerEnable (unsigned int baseAdd, unsigned int dataFormat)

  - *It Enables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).*

- void SPICSTimerDisable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Disables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).*

- void SPICharLengthSet (unsigned int baseAdd, unsigned int flag, unsigned int dataFormat)
  - *It Set the Charcter length.*

- void SPIShiftMsbFirst (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI to Transmit MSB bit first during Data transfer.*
- void SPIShiftLsbFirst (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI to Transmit LSB bit first during Data transfer.*
- void SPIParityEnable (unsigned int baseAdd, unsigned int flag, unsigned int dataFormat)
  - *It Enables Parity in SPI and also configures Even or Odd Parity.*

- void SPIParityDisable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Disables Parity in SPI.*

- void SPIWdelaySet (unsigned int baseAdd, unsigned int flag, unsigned int dataFormat)
  - *It sets the Delay between SPI transmission.*

- void SPIWaitEnable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI Master to wait for SPIx_ENA signal.*

- void SPIWaitDisable (unsigned int baseAdd, unsigned int dataFormat)
  - *It Configures SPI Master not to wait for SPIx_ENA signal.*

- void SPIIntLevelSet (unsigned int baseAdd, unsigned int flag)
  - *It Configures SPI to Map interrupts to interrupt line INT1.*

- void SPIIntEnable (unsigned int baseAdd, unsigned int flag)
  - *It Enables the interrupts.*
- void SPIIntDisable (unsigned int baseAdd, unsigned int flag)
  - *It Disables the interrupts.*
- void SPIIntStatusClear (unsigned int baseAdd, unsigned int flag)
  - *It clears Status of interrupts.*
- unsigned int SPIIntStatus (unsigned int baseAdd, unsigned int flag)
  - *It reads the Status of interrupts.*
- unsigned int SPIDataReceive (unsigned int baseAdd)
  - *It recevies data by reading from SPIBUF register.*

- unsigned int SPIInterruptVectorGet (unsigned int baseAdd)
  - *It returns the vector of the pending interrupt at interrupt line INT1.*

### 4.46.1 Detailed Description

SPI APIs and macros.

This file contains the driver API prototypes and macro definitions.

### 4.46.2 Function Documentation

#### 4.46.2.1 void SPICharLengthSet ( unsigned int *baseAdd,* unsigned int *numOfChar,* unsigned int *dataFormat* )

- It Set the Charcter length.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value which determines the number of the charcters to be transmitted . |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.46.2.2 void SPIClkConfigure ( unsigned int *baseAdd,* unsigned int *moduleClk,* unsigned int *spiClk,* unsigned int *dataFormat* )

- It will configure the prescalar to generate required spi clock
.

**Parameters**

| | |
|---|---|
| - | baseAdd is Memory Address of the SPI instance used<br>. |
| - | moduleClk is the input clk to SPI module from PLL<br>. |
| - | spiClk is the spi bus speed<br>. |
| - | dataFormat is instance of the data format register used<br>. |

**Returns**

none.

#### 4.46.2.3 void SPIConfigClkFormat ( unsigned int *baseAdd,* unsigned int *flag,* unsigned int *dataFormat* )

- It Configures SPI Clock's Phase and Polarity.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value which determines Phase and Polarity of the Clock..<br>flag can take following values.<br>SPI_CLK_POL_HIGH - Clock is High Before and after data transfer. SPI_CLK_POL_LOW - Clock is Low Before and after data transfer. SPI_CLK_INPHASE - Clock is not Delayed. SPI_CLK_OUTOFPHASE - Clock is Delayed by half clock cycle. |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.46.2.4 void SPICSTimerDisable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Disables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>`dataFormat can take following value.\n`<br><br>`SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n`<br>`SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n`<br>`SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n`<br>`SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n` |

**Returns**

none.

#### 4.46.2.5 void SPICSTimerEnable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Enables Transmit-end-to-chip-select-inactive-delay(t2cdelay) and Chip-select-active-to-transmit-start-delay(c2tdelay).

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.46.2.6 void SPIDat1Config ( unsigned int *baseAdd,* unsigned int *flag,* unsigned char *cs* )

- Configures the SPIDat1 Register.It won't write to TX part of the SPIDat1 Register.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI data instance used. |
| - | flag is value to Configure CSHOL,Wait Delay Conter Enable bit and to select the appropriate DataFormat register.<br>flag can take following values.<br>SPI_CSHOLD - To Hold the CS line active after data Transfer untill new data and Control information is loaded.<br>SPI_DELAY_COUNTER_ENA - Enables Delay Counter.<br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3. |
| - | cs is the value to driven on CS pin(line). |

**Returns**

none.

#### 4.46.2.7 unsigned int SPIDataReceive ( unsigned int *baseAdd* )

- It recevies data by reading from SPIBUF register.

**Parameters**

| | |
|---|---|
| - | baseAdd is the memory instance to be used. |

**Returns**

received data.

#### 4.46.2.8 void SPIDefaultCSSet ( unsigned int *baseAdd,* unsigned char *dcsval* )

- Sets the default value for CS pin(line) when no transmission is is performed by writing to SPIDEF Reg.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | dcsval is the value written to SPIDEF register to set default value on CS pin(line). |

**Returns**

none.

#### 4.46.2.9 void SPIDelayConfigure ( unsigned int *baseAdd,* unsigned int *c2edelay,* unsigned int *t2edelay,* unsigned int *t2cdelay,* unsigned int *c2tdelay* )

- Configures SPI CS and ENA Delay in SPIDELAY Register.

**Parameters**

| | |
|---|---|
| - | baseAdd is Memory Address of the SPI instance. |
| - | c2edelay is the Chip-select-active-to-SPIx_ENA-signal -active-time-out. |
| - | t2edelay is the Transmit-data-finished-to-SPIx_ENA-pin -inactive-time-out. |
| - | t2cdelay is the Transmit-end-to-chip-select-inactive-delay. |
| - | c2tdelay is the Chip-select-active-to-transmit-start-delay. |

**Returns**

none.

Note: SPIx_CS and SPI_ENA are active low pins.

#### 4.46.2.10 void SPIEnable ( unsigned int *baseAdd* )

- It will Enable SPI module

.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used<br>. |

**Returns**

none.

#### 4.46.2.11 void SPIIntDisable ( unsigned int *baseAdd,* unsigned int *flag* )

- It Disables the interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts required to be Enabled.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n |

**Returns**

none.

#### 4.46.2.12 void SPIIntEnable ( unsigned int *baseAdd,* unsigned int *flag* )

- It Enables the interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts required to be Enabled.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n |

**Returns**

none.

#### 4.46.2.13 unsigned int SPIInterruptVectorGet ( unsigned int *baseAdd* )

- It returns the vector of the pending interrupt at interrupt line INT1.

**Parameters**

| | |
|---|---|
| - | baseAdd is the memory instance to be used. |

**Returns**

vector of the pending interrupt at interrupt line INT1.

#### 4.46.2.14 void SPIIntLevelSet ( unsigned int *baseAdd,* unsigned int *flag* )

- It Configures SPI to Map interrupts to interrupt line INT1.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts required to be mapped.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INTLVL – Data length error interrupt level.\n<br>SPI_TIMEOUT_INTLVL – TimeOut length error interrupt level.\n<br>SPI_PARITY_ERR_INTLVL – Parity error interrupt level.\n<br>SPI_DESYNC_SLAVE_INTLVL – Desyncrozied slave interrupt level.\n<br>SPI_BIT_ERR_INTLVL – Bit error interrupt level.\n<br>SPI_RECV_OVERRUN_INTLVL – Receive Overrun interrupt level.\n<br>SPI_RECV_INTLVL – Receive interrupt level.\n<br>SPI_TRANSMIT_INTLVL – Transmit interrupt level.\n |

**Returns**

none.

#### 4.46.2.15 unsigned int SPIIntStatus ( unsigned int *baseAdd,* unsigned int *flag* )

- It reads the Status of interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts whose status needs to be read.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n<br>SPI_DMA_REQUEST_ENA_INT – DMA request interrupt.\n |

**Returns**

status of interrupt.

#### 4.46.2.16 void SPIIntStatusClear ( unsigned int *baseAdd,* unsigned int *flag* )

- It clears Status of interrupts.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | flag is the interrupts whose status needs to be cleared.<br><br>flag can take following values.\n<br><br>SPI_DATALEN_ERR_INT – Data length error interrupt.\n<br>SPI_TIMEOUT_INT – TimeOut length error interrupt.\n<br>SPI_PARITY_ERR_INT – Parity error interrupt.\n<br>SPI_DESYNC_SLAVE_INT – Desyncrozied slave interrupt.\n<br>SPI_BIT_ERR_INT – Bit error interrupt.\n<br>SPI_RECV_OVERRUN_INT – Receive Overrun interrupt.\n<br>SPI_RECV_INT – Receive interrupt.\n<br>SPI_TRANSMIT_INT – Transmit interrupt.\n<br>SPI_DMA_REQUEST_ENA_INT – DMA request interrupt.\n |

**Returns**

none.

#### 4.46.2.17 void SPIModeConfigure ( unsigned int *baseAdd,* unsigned int *flag* )

- Configures SPI to master or slave mode.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |

**Returns**

none.

#### 4.46.2.18 void SPIOutOfReset ( unsigned int *baseAdd* )

- Brings SPI out of reset state.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |

**Returns**

none.

#### 4.46.2.19 void SPIParityDisable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Disables Parity in SPI.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.46.2.20 void SPIParityEnable ( unsigned int *baseAdd,* unsigned int *flag,* unsigned int *dataFormat* )

- It Enables Parity in SPI and also configures Even or Odd Parity.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value determines whether odd or even Parity.<br><br>flag can take following values.\n<br><br>SPI_ODD_PARITY - selects odd parity<br>SPI_EVEN_PARITY - selects even parity |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.46.2.21 void SPIPinControl ( unsigned int *baseAdd,* unsigned int *idx,* unsigned int * *val* )

- Configures SPI Pin Control Registers.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |
| - | idx is the Pin Control register number.It can take any integer value between 0 and 5. |
| - | flag is to indicate to whether to (1)read from Pin Control Register or to (0)write to Pin Control Register. |

| | |
|---|---|
| - | val is a value/return argument which has the value to be written in case of writes or the value read in case of reads |

**Returns**

none.

#### 4.46.2.22 void SPIReset ( unsigned int *baseAdd* )

- It will put SPI in to reset state.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory Address of the SPI instance used. |

**Returns**

none.

#### 4.46.2.23 void SPIShiftLsbFirst ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI to Transmit LSB bit first during Data transfer.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.46.2.24 void SPIShiftMsbFirst ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI to Transmit MSB bit first during Data transfer.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

#### 4.46.2.25 void SPITransmitData1 ( unsigned int *baseAdd,* unsigned int *data* )

- Trasmits Data from TX part of SPIDAT1 register.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the SPI instance used. |
| - | data is the data transmitted out of SPI. |

**Returns**

none.

#### 4.46.2.26 void SPIWaitDisable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI Master not to wait for SPIx_ENA signal.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

Note:It is applicable only in SPI Master Mode.SPIx_ENA is a active low signal

#### 4.46.2.27 void SPIWaitEnable ( unsigned int *baseAdd,* unsigned int *dataFormat* )

- It Configures SPI Master to wait for SPIx_ENA signal.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

**Returns**

none.

Note:It is applicable only in SPI Master Mode.SPIx_ENA is a active low signal

#### 4.46.2.28 void SPIWdelaySet ( unsigned int *baseAdd,* unsigned int *flag,* unsigned int *dataFormat* )

- It sets the Delay between SPI transmission.

**Parameters**

| | |
|---|---|
| - | baseAdd is the Memory address of the the SPI instance used. |
| - | flag is the value determines amount of delay. |
| - | dataFormat is the value to select the Format register.<br><br>dataFormat can take following value.\n<br><br>SPI_DATA_FORMAT0 - To select DataFormat Register 0.\n<br>SPI_DATA_FORMAT1 - To select DataFormat Register 1.\n<br>SPI_DATA_FORMAT2 - To select DataFormat Register 2.\n<br>SPI_DATA_FORMAT3 - To select DataFormat Register 3.\n |

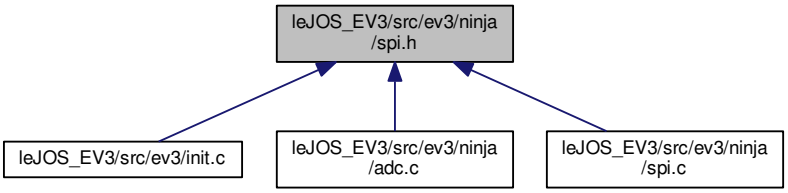
**Returns**

none.

## 4.47 leJOS_EV3/src/ev3/ninja/spi.h File Reference

This header declares function required to interact with the SPI0 controller of the SoC which is connected to the ADC.

This graph shows which files directly or indirectly include this file:




### Functions

- unsigned short spi_update (unsigned short data)

  *Send data via the SPI0 controller and get an updated value received by the SPI0 controller.*
- void spi_init (void)

  *Initialize the SPI0 controller and the required GPIO pins.*

### 4.47.1 Detailed Description

This header declares function required to interact with the SPI0 controller of the SoC which is connected to the ADC.

**Author**

ev3ninja

### 4.47.2 Function Documentation

#### 4.47.2.1 void spi_init ( void )

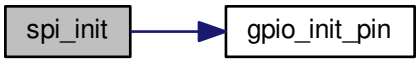
Initialize the SPI0 controller and the required GPIO pins.

For the GPIO pins, the SPI functionality will be set instead of the GPIO functionality.

**Returns**

none

Here is the call graph for this function:




#### 4.47.2.2 unsigned short spi_update ( unsigned short *data* )

Send data via the SPI0 controller and get an updated value received by the SPI0 controller.

Since the SPI0 controller is configured for chip select 3 only, all data sent and received will be to or from the ADC. According to the documentation of the ADC, the selected channel (0 to 15) will be probed and returned on frame n+2. Therefore, the command is sent 3 times and only the third value received is returned to the caller of this function.

**Parameters**

| | |
|---|---|
| *data* | - The data to send (since this is data to the ADC, check the ADC documentation for possible commands that can be sent) |

**Returns**

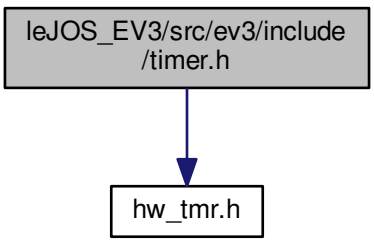
The value received from the ADC after sending the data 3 times

## 4.48 leJOS_EV3/src/ev3/include/timer.h File Reference

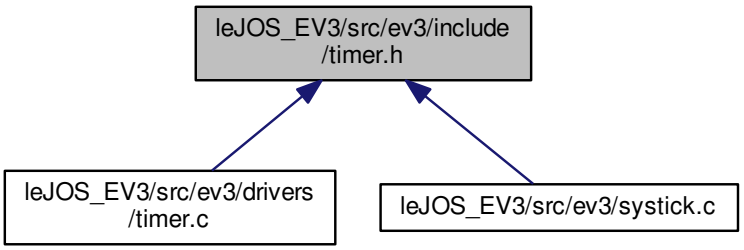
Timer APIs and macros.

```
#include "hw_tmr.h"
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:




## Macros

- #define **TMR_TIMER12** (0x00003FFEu) /∗ Timer12 ∗/
- #define **TMR_TIMER34** (0x3FFE0000u) /∗ Timer34 ∗/
- #define **TMR_TIMER_BOTH** (0x3FFE3FFEu) /∗ Both Timers ∗/
- #define **TMR_ENABLE_ONCE** (0x00400040u)
- #define **TMR_ENABLE_CONT** (0x00800080u)
- #define **TMR_ENABLE_CONTRELOAD** (0x00C000C0u)
- #define **TMR_CFG_64BIT_CLK_INT** (0x00000013u)
- #define **TMR_CFG_64BIT_CLK_EXT** (0x01000113u)
- #define **TMR_CFG_64BIT_WATCHDOG** (0x0000001Bu)
- #define **TMR_CFG_32BIT_CH_CLK_INT** (0x0000001Fu)
- #define **TMR_CFG_32BIT_CH_CLK_EXT** (0x0100011Fu)
- #define **TMR_CFG_32BIT_UNCH_CLK_BOTH_INT** (0x00000017u)
- #define **TMR_CFG_32BIT_UNCH_CLK_12INT_34EXT** (0x01000017u)
- #define **TMR_CFG_32BIT_UNCH_CLK_12EXT_34INT** (0x00000117u)
- #define **TMR_CFG_32BIT_UNCH_CLK_BOTH_EXT** (0x01000117u)
- #define **TMR_INT_TMR12_CAPT_MODE** (TMR_INTCTLSTAT_EVTINTEN12)
- #define **TMR_INT_TMR12_NON_CAPT_MODE** (TMR_INTCTLSTAT_PRDINTEN12)
- #define **TMR_INT_TMR34_CAPT_MODE** (TMR_INTCTLSTAT_EVTINTEN34)
- #define **TMR_INT_TMR34_NON_CAPT_MODE** (TMR_INTCTLSTAT_PRDINTEN34)
- #define **TMR_INTSTAT12_TIMER_NON_CAPT** (TMR_INTCTLSTAT_PRDINTSTAT12)
- #define **TMR_INTSTAT12_TIMER_CAPT** (TMR_INTCTLSTAT_EVTINTSTAT12)
- #define **TMR_INTSTAT34_TIMER_NON_CAPT** (TMR_INTCTLSTAT_PRDINTSTAT34)
- #define **TMR_INTSTAT34_TIMER_CAPT** (TMR_INTCTLSTAT_EVTINTSTAT12)
- #define **TMR_CAPT_DISABLE** (0x00000000)
- #define **TMR_CAPT_ENABLE_RIS_EDGE** (0x08000800)
- #define **TMR_CAPT_ENABLE_FALL_EDGE** (0x18001800)
- #define **TMR_CAPT_ENABLE_BOTH_EDGE** (0x28002800)
- #define **TMR_PULSE_WIDTH_1_CLK** (0x00000000) /∗ 1 clock cycle ∗/
- #define **TMR_PULSE_WIDTH_2_CLK** (0x00100010) /∗ 2 clock cycles ∗/
- #define **TMR_PULSE_WIDTH_3_CLK** (0x00200020) /∗ 3 clock cycles ∗/
- #define **TMR_PULSE_WIDTH_4_CLK** (0x00300030) /∗ 4 clock cycles ∗/
- #define **TMR_OUT12_ASSERTED** (TMR_TCR_TSTAT12) /∗ TMR64P_OUT12 ∗/
- #define **TMR_OUT34_ASSERTED** (TMR_TCR_TSTAT34) /∗ TMR64P_OUT34 ∗/

## Functions

- unsigned int TimerCompareGet (unsigned int baseAddr, unsigned int regIndex)

  *Returns the Compare value of the specified CMP register.*
- unsigned int TimerOUTStatusGet (unsigned int baseAddr, unsigned int timer)

  *Returns the timer status. The timer status Drives the value of the timer output TM64P_OUTn when configured.*
- unsigned int TimerIntStatusGet (unsigned int baseAddr, unsigned int statFlag)

  *Returns the status of specified timer interrupts.*
- unsigned int TimerIntStatusClear (unsigned int baseAddr, unsigned int statFlag)

  *Clears the status of specified timer interrupts.*
- unsigned int TimerCaptureGet (unsigned int baseAddr, unsigned int timer)

  *Returns the capture value of the specified timer.*
- unsigned int TimerCounterGet (unsigned int baseAddr, unsigned int timer)

  *Returns the Counter register contents of the specified timer.*
- unsigned int TimerPeriodGet (unsigned int baseAddr, unsigned int timer)

  *Returns the Period register contents of the specified timer.*
- unsigned int TimerReloadGet (unsigned int baseAddr, unsigned int timer)

  *Returns the Reload Period of the specified timer.*
- unsigned int TimerPreScalarCount34Get (unsigned int baseAddr)

  *Returns the Prescalar Counter of Timer34.*
- unsigned int TimerDivDwnRatio34Get (unsigned int baseAddr)

  *returns the Timer Divide Down Ratio of Timer34.*
- void TimerCounterSet (unsigned int baseAddr, unsigned int timer, unsigned int counter)

  *Set the Counter register of the specified timer.*
- void TimerPeriodSet (unsigned int baseAddr, unsigned int timer, unsigned int period)

  *Set the Period register(s) of the specified timer(s).*
- void TimerReloadSet (unsigned int baseAddr, unsigned int timer, unsigned int reload)

  *Set the Reload period of the specified timer.*
- void TimerEnable (unsigned int baseAddr, unsigned int timer, unsigned int enaMode)

  *Enables the timer in the specified mode. The timer must be configured before it is enabled. The timer starts running when this API is called.*
- void TimerPreScalarCount34Set (unsigned int baseAddr, unsigned int psc34)

  *Sets the Prescalar Counter of Timer34.*
- void TimerDivDwnRatio34Set (unsigned int baseAddr, unsigned int tddr34)

  *Sets the Timer Divide Down Ratio of Timer34.*
- void TimerInvertOUTDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the inversion the TMR64P_OUTn signal.*
- void TimerReadResetDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the timer for read reset mode.*
- void TimerInputGateDisable (unsigned int baseAddr, unsigned int timer)

  *Disable the Input Gate. Timer clock will not be gated by input pin.*
- void TimerReadResetEnable (unsigned int baseAddr, unsigned int timer)

  *Enables the timer(s) for read reset mode. The timer shall be Configured in 32 bit unchained mode before this API is called. Read reset determines the effect of timer counter read on TIMn. If Read reset is enabled, the timer counter will be reset when the timer counter register TIMn is read.*
- void TimerInvertINDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the Inversion of the TMR64P_INn signal.*
- void TimerInvertOUTEnable (unsigned int baseAddr, unsigned int timer)

  *Inverts the TMR64P_OUTn signal.*
- void TimerInputGateEnable (unsigned int baseAddr, unsigned int timer)

  *Sets the Input Gate Enable. Allows the timer to gate the internal timer clock source. The timer starts counting when the input pin goes from Low to High and stops counting when transition happens from high to low.*

- void TimerInvertINEnable (unsigned int baseAddr, unsigned int timer)

  *Inverts the TMR64P_INn signal.*
- void TimerIntDisable (unsigned int baseAddr, unsigned int intFlags)

  *Disables the specified timer interrupts.*
- void TimerClockModeSet (unsigned int baseAddr, unsigned int timer)

  *Sets the clock mode. Once the clock mode is set, the outpin behaves as 50% duty cycle signal.*
- void TimerPulseModeSet (unsigned int baseAddr, unsigned int timer)

  *Sets the pulse mode. The outpin goes active when the timer count reaches the period.*
- void TimerIntEnable (unsigned int baseAddr, unsigned int intFlags)

  *Enables the specified timer interrupts. The timer interrupts which are to be enabled can be passed as parameter to this function.*
- void TimerConfigure (unsigned int baseAddr, unsigned int config)

  *Configures the timer. The timer can be configured in 64 bit mode 32 bit chained/unchained mode, or as a watchdog timer. The timer can be given external clock input or internal clock input. When this API is called,*
  *> The Timer counters are cleared*
  *> Both the timers are disabled from Reset. Hence, both the timers will start counting when enabled.*
  *.*
- void TimerDisable (unsigned int baseAddr, unsigned int timer)

  *Disables the timer. The timer stops running when this API is called.*
- void TimerWatchdogReactivate (unsigned int baseAddr)

  *Re-activate the Watchdog timer. The WDT shall be enabled before this API is called. The user shall call this API before the WDT expires, to avoid a reset.*
- void TimerWatchdogActivate (unsigned int baseAddr)

  *Activate the Watchdog timer. The timer shall be configured as watchdog timer before this API is called. This API writes two keys into the WDTCR in the order to activate the WDT. The user shall call TimerWatchdogReactivate API before the WDT expires, to avoid a reset.*
- void TimerCaptureConfigure (unsigned int baseAddr, unsigned int timer, unsigned int cfgCap)

  *Configures the Timer for Capture Mode. The Timer Module Shall be Configured in 32 bit unchained mode before this API is called.*
- void TimerCompareSet (unsigned int baseAddr, unsigned int regIndex, unsigned int compare)

  *Set the Compare value of the specified CMP register.*
- void TimerPulseWidthSet (unsigned int baseAddr, unsigned int timer, unsigned int pulseWidth)

  *Sets the pulse width for the specified timer. Determines the pulse. width in the TSTATn bit and the OUT pin in pulse mode.*

### 4.48.1 Detailed Description

Timer APIs and macros.

This file contains the driver API prototypes and macro definitions.

### 4.48.2 Function Documentation

#### 4.48.2.1 void TimerCaptureConfigure ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *cfgCap* )

Configures the Timer for Capture Mode. The Timer Module Shall be Configured in 32 bit unchained mode before this API is called.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

| | |
|---|---|
| *timer* | The timer, of which capture to be configured. |
| *cfgCap* | Configuration of Capture Mode. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers
cfgCap can take the values
TMR_CAPT_DISABLE - Capture Mode disable
TMR_CAPT_ENABLE_RIS_EDGE - Capture enable at rising edge
TMR_CAPT_ENABLE_FALL_EDGE - Capture enable at falling edge
TMR_CAPT_ENABLE_BOTH_EDGE - Capture enable at both edges

**Returns**

None.

#### 4.48.2.2 unsigned int TimerCaptureGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the capture value of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which capture value to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Capture Value

#### 4.48.2.3 void TimerClockModeSet ( unsigned int *baseAddr,* unsigned int *timer* )

Sets the clock mode. Once the clock mode is set, the outpin behaves as 50% duty cycle signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which clock mode to be set. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.4 unsigned int TimerCompareGet ( unsigned int *baseAddr,* unsigned int *regIndex* )

Returns the Compare value of the specified CMP register.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *regIndex* | Index of the CMP Register |

regIndex can take any value from 0 to 7. If regIndex = n, contents of CMPn will be returned.

**Returns**

Compare Value.

#### 4.48.2.5 void TimerCompareSet ( unsigned int *baseAddr,* unsigned int *regIndex,* unsigned int *compare* )

Set the Compare value of the specified CMP register.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *regIndex* | Index of the CMP Register |
| *compare* | The value to be written |

regIndex can take any value from 0 to 7. If regIndex = n, CMPn will be set with the value compare.

**Returns**

None.

#### 4.48.2.6 void TimerConfigure ( unsigned int *baseAddr,* unsigned int *config* )

Configures the timer. The timer can be configured in 64 bit mode 32 bit chained/unchained mode, or as a watchdog timer. The timer can be given external clock input or internal clock input. When this API is called,
> The Timer counters are cleared
> Both the timers are disabled from Reset. Hence, both the timers will start counting when enabled.
.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *config* | Configuration of the Timer Module. |

config can take the values
TMR_CFG_64BIT_CLK_INT - 64 bit mode with internal clock
TMR_CFG_64BIT_CLK_EXT - 64 bit mode with external clock
TMR_CFG_64BIT_WATCHDOG - 64 bit watchdog timer mode
TMR_CFG_32BIT_CH_CLK_INT - 32 bit chained mode with internal clock
TMR_CFG_32BIT_CH_CLK_EXT - 32 bit chained mode with external clock
TMR_CFG_32BIT_UNCH_CLK_BOTH_INT - 32 bit unchained mode; Both timers clock sources are internal
TMR_CFG_32BIT_UNCH_CLK_12INT_34EXT - 32 bit unchained mode; Clock source for Timer12 is internal and for Timer34 is external
TMR_CFG_32BIT_UNCH_CLK_12EXT_34INT - 32 bit unchained mode; Clock source for Timer12 is external and for Timer34 is internal
TMR_CFG_32BIT_UNCH_CLK_BOTH_EXT - 32 bit unchained mode; Both timers clock sources are external

**Returns**

None.

#### 4.48.2.7 unsigned int TimerCounterGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the Counter register contents of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which counter to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Counter Value.

#### 4.48.2.8 void TimerCounterSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *counter* )

Set the Counter register of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which counter to be set. |
| *counter* | The counter value of the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.9 void TimerDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the timer. The timer stops running when this API is called.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | Timer to be disabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.10 unsigned int TimerDivDwnRatio34Get ( unsigned int *baseAddr* )

returns the Timer Divide Down Ratio of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

TDDR34 value.

#### 4.48.2.11 void TimerDivDwnRatio34Set ( unsigned int *baseAddr,* unsigned int *tddr34* )

Sets the Timer Divide Down Ratio of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *tddr34* | TDDR34, Timer Divide Down Ratio |

**Returns**

None.

#### 4.48.2.12 void TimerEnable ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *enaMode* )

Enables the timer in the specified mode. The timer must be configured before it is enabled. The timer starts running when this API is called.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer to be enabled. |
| *enaMode* | Mode of enabling the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer 12 only
TMR_TIMER_BOTH - Both timers
enaMode can take the values
TMR_ENABLE_ONCE - Enable the timer to run once
TMR_ENABLR_CONT - Enable to run continuous
TMR_ENABLE_CONTRELOAD - Enable to run continuous with period reload

**Returns**

None.

#### 4.48.2.13 void TimerInputGateDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disable the Input Gate. Timer clock will not be gated by input pin.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which input gate to be disabled. |

timer can take the values

TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.14 void TimerInputGateEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Sets the Input Gate Enable. Allows the timer to gate the internal timer clock source. The timer starts counting when the input pin goes from Low to High and stops counting when transition happens from high to low.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which input gate to be enabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.15 void TimerIntDisable ( unsigned int *baseAddr,* unsigned int *intFlags* )

Disables the specified timer interrupts.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *intFlags* | Timer Interrupts to be disabled |

intFlags can take any, or a combination of the following values
TMR_INT_TMR12_CAPT_MODE - Disable Timer12 interrupt in Capture Mode
TMR_INT_TMR12_NON_CAPT_MODE - Disable Timer12 interrupt in normal mode
TMR_INT_TMR34_CAPT_MODE - Disable Timer34 interrupt in Capture mode
TMR_INT_TMR34_NON_CAPT_MODE - Disable Timer34 interrupt in normal mode

**Returns**

None.

#### 4.48.2.16 void TimerIntEnable ( unsigned int *baseAddr,* unsigned int *intFlags* )

Enables the specified timer interrupts. The timer interrupts which are to be enabled can be passed as parameter to this function.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *intFlags* | Timer Interrupts to be enabled |

intFlags can take any, or a combination of the following values
TMR_INT_TMR12_CAPT_MODE - Enable Timer12 interrupt in Capture Mode
TMR_INT_TMR12_NON_CAPT_MODE - Enable Timer12 interrupt in normal mode
TMR_INT_TMR34_CAPT_MODE - Enable Timer34 interrupt in Capture mode
TMR_INT_TMR34_NON_CAPT_MODE - Enable Timer34 interrupt in normal mode

**Returns**

None.

#### 4.48.2.17 unsigned int TimerIntStatusClear ( unsigned int *baseAddr,* unsigned int *statFlag* )

Clears the status of specified timer interrupts.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *statFlag* | Status flags to be cleared. |

intFlags can take any or combination of the following values
TMR_INTSTAT12_TIMER_NON_CAPT - Timer12 interrupt status in normal mode
TMR_INTSTAT12_TIMER_CAPT - Timer12 interrupt status in capture mode
TMR_INTSTAT34_TIMER_NON_CAPT - Timer34 interrupt status in normal mode
TMR_INTSTAT34_TIMER_CAPT - Timer34 interrupt status in capture mode

**Returns**

None

#### 4.48.2.18 unsigned int TimerIntStatusGet ( unsigned int *baseAddr,* unsigned int *statFlag* )

Returns the status of specified timer interrupts.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *statFlag* | Status flags to be read. |

intFlags can take any or combination of the following values
TMR_INTSTAT12_TIMER_NON_CAPT - Timer12 interrupt status in normal mode
TMR_INTSTAT12_TIMER_CAPT - Timer12 interrupt status in capture mode
TMR_INTSTAT34_TIMER_NON_CAPT - Timer34 interrupt status in normal mode
TMR_INTSTAT34_TIMER_CAPT - Timer34 interrupt status in capture mode

**Returns**

Status of Interrupt. Returns all the fields of which status is set

Note : This API will return the same fields which is passed as parameter, if all the specified interrupt status is set. The return value will be 0 if none of the interrupt status in the parameter passed is set.

#### 4.48.2.19 void TimerInvertINDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the Inversion of the TMR64P_INn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be disabled. |

timer can take the values
TMR_TIMER34 - TMR64P_INT34 inversion will be disabled
TMR_TIMER12 - TMR64P_IN12 inversion will be disabled
TMR_TIMER_BOTH - Both TMR64P_IN12 and TMR64P_IN34 inversion disabled

**Returns**

None.

#### 4.48.2.20 void TimerInvertINEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Inverts the TMR64P_INn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be enabled. |

timer can take the following values.
TMR_TIMER34 - Inverts TMR64P_IN34 signal
TMR_TIMER12 - Inverts TMR64P_IN12 signal
TMR_TIMER_BOTH - Inverts both TMR64P_IN12 and TMR64P_IN34 signals

**Returns**

None.

#### 4.48.2.21 void TimerInvertOUTDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the inversion the TMR64P_OUTn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be disabled. |

timer can take the values
TMR_TIMER34 - TMR64P_OUT34 inversion will be disabled
TMR_TIMER12 - TMR64P_OUT12 inversion will be disabled
TMR_TIMER_BOTH - Both TMR64P_OUT12 and TMR64P_OUT34 inversion disabled

**Returns**

None.

#### 4.48.2.22 void TimerInvertOUTEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Inverts the TMR64P_OUTn signal.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which inversion to be enabled. |

timer can take the values
TMR_TIMER34 - Inverts TMR64P_OUT34 signal
TMR_TIMER12 - Inverts TMR64P_OUT12 signal
TMR_TIMER_BOTH - Inverts both TMR64P_OUT12 and TMR64P_OUT34 signals

**Returns**

None.

#### 4.48.2.23 unsigned int TimerOUTStatusGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the timer status. The timer status Drives the value of the timer output TM64P_OUTn when configured.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which status to be read |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12
TMR_TIMER_BOTH - Both timers
Note : This API returns 0 if none of the status bits is set.

**Returns**

Status of the timer. Returns the following values or the combination of both.
TMR_OUT12_ASSERTED - TMR64P_OUT12 is asserted
TMR_OUT34_ASSERTED - TMR64P_OUT34 is asserted

#### 4.48.2.24 unsigned int TimerPeriodGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the Period register contents of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which period to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Period Value

#### 4.48.2.25 void TimerPeriodSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *period* )

Set the Period register(s) of the specified timer(s).

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which period to be set. |
| *period* | The period value of the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.26 unsigned int TimerPreScalarCount34Get ( unsigned int *baseAddr* )

Returns the Prescalar Counter of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

Prescalar Value.

#### 4.48.2.27 void TimerPreScalarCount34Set ( unsigned int *baseAddr,* unsigned int *psc34* )

Sets the Prescalar Counter of Timer34.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *psc34* | 4 bit prescalar value |

**Returns**

None.

#### 4.48.2.28 void TimerPulseModeSet ( unsigned int *baseAddr,* unsigned int *timer* )

Sets the pulse mode. The outpin goes active when the timer count reaches the period.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which pulse mode to be set. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.29 void TimerPulseWidthSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *pulseWidth* )

Sets the pulse width for the specified timer. Determines the pulse. width in the TSTATn bit and the OUT pin in pulse mode.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which pulse width to be set. |
| *pulseWidth* | Pulse width to be set. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers
pulseWidth can take the following values
TMR_PULSE_WIDTH_1_CLK - 1 clock cycle
TMR_PULSE_WIDTH_2_CLK - 2 clock cycles
TMR_PULSE_WIDTH_3_CLK - 3 clock cycles
TMR_PULSE_WIDTH_4_CLK - 4 clock cycles

**Returns**

None.

#### 4.48.2.30 void TimerReadResetDisable ( unsigned int *baseAddr,* unsigned int *timer* )

Disables the timer for read reset mode.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which Read Reset to be disabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.31 void TimerReadResetEnable ( unsigned int *baseAddr,* unsigned int *timer* )

Enables the timer(s) for read reset mode. The timer shall be Configured in 32 bit unchained mode before this API is called. Read reset determines the effect of timer counter read on TIMn. If Read reset is enabled, the timer counter will be reset when the timer counter register TIMn is read.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which Read Reset to be enabled. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.32 unsigned int TimerReloadGet ( unsigned int *baseAddr,* unsigned int *timer* )

Returns the Reload Period of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which Reload value to be read. |

timer can take the values
TMR_TIMER34 - Timer34
TMR_TIMER12 - Timer12

**Returns**

Reload Value

#### 4.48.2.33 void TimerReloadSet ( unsigned int *baseAddr,* unsigned int *timer,* unsigned int *reload* )

Set the Reload period of the specified timer.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |
| *timer* | The timer, of which reload period to be set. |
| *reload* | The reload period value of the timer. |

timer can take the values
TMR_TIMER34 - Timer34 only
TMR_TIMER12 - Timer12 only
TMR_TIMER_BOTH - Both timers

**Returns**

None.

#### 4.48.2.34 void TimerWatchdogActivate ( unsigned int *baseAddr* )

Activate the Watchdog timer. The timer shall be configured as watchdog timer before this API is called. This API writes two keys into the WDTCR in the order to activate the WDT. The user shall call TimerWatchdogReactivate API before the WDT expires, to avoid a reset.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

None.

#### 4.48.2.35 void TimerWatchdogReactivate ( unsigned int *baseAddr* )

Re-activate the Watchdog timer. The WDT shall be enabled before this API is called. The user shall call this API before the WDT expires, to avoid a reset.

**Parameters**

| | |
|---|---|
| *baseAddr* | Base Address of the Timer Module Registers. |

**Returns**

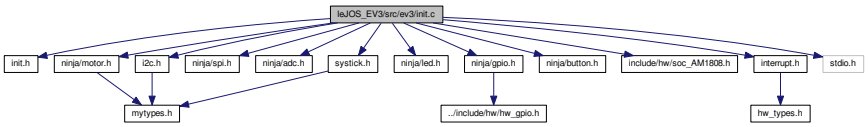
None.

## 4.49 leJOS_EV3/src/ev3/init.c File Reference

This code is required in order to initialize the leJOS driver including all modules properly.

```
#include "init.h"
#include "systick.h"
#include "i2c.h"
#include "ninja/spi.h"
#include "ninja/adc.h"
#include "ninja/gpio.h"
#include "ninja/led.h"
#include "ninja/motor.h"
#include "ninja/button.h"
#include "include/hw/soc_AM1808.h"
#include "interrupt.h"
#include "stdio.h"
```

Include dependency graph for init.c:




### Functions

- void interrupt_init (void)

  *Initialize the AINTC by enabling normal interrupts (IRQ) and fast interrupts (FIQ) on all required levels.*
- void sensor_init (void)

  *Initialize the sensor ports of the EV3 by setting the correct pin-multiplexing configuration.*
- void leJOS_init (void)

  *Initialize the leJOS driver and all required submodules.*

### Variables

- volatile U8 is_AINTC_initialized = 0

  *A flag indicating if the AINTC (Interrupt Controller) is already initialized or not.*
- sensor_port_info ports [ ]

  *Array storing information about the 4 sensor ports of the EV3.*

### 4.49.1 Detailed Description

This code is required in order to initialize the leJOS driver including all modules properly.

**Author**

Tobias Schießl, ev3ninja

### 4.49.2 Function Documentation

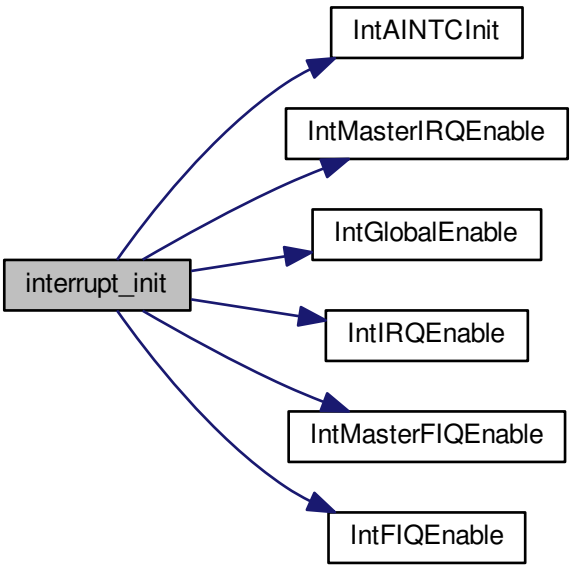
#### 4.49.2.1 void interrupt_init ( void )

Initialize the AINTC by enabling normal interrupts (IRQ) and fast interrupts (FIQ) on all required levels.

**Returns**

none

Here is the call graph for this function:




#### 4.49.2.2 void leJOS_init ( void )

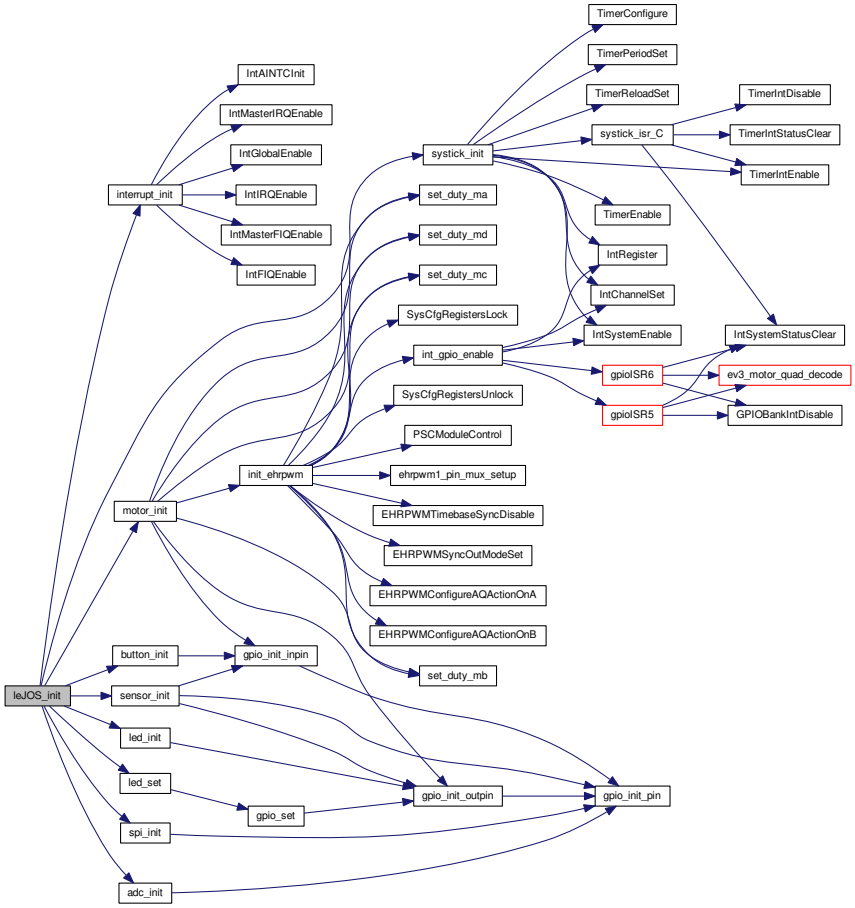
Initialize the leJOS driver and all required submodules.

Initialization contains: AINTC, systick, sensor ports, motor ports, SPI controller, ADC, hardware buttons, LEDs

**Returns**

none

Here is the call graph for this function:



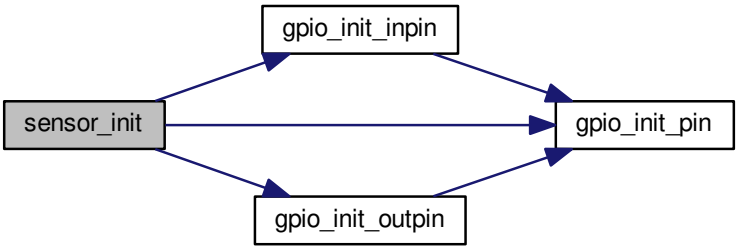

#### 4.49.2.3 void sensor_init ( void )

Initialize the sensor ports of the EV3 by setting the correct pin-multiplexing configuration.

**Returns**

none

Here is the call graph for this function:




### 4.49.3 Variable Documentation

#### 4.49.3.1 volatile U8 is_AINTC_initialized = 0

A flag indicating if the AINTC (Interrupt Controller) is already initialized or not.

This is required since the function leJOS_init will be called twice in OSEK applications, but the systick and the AINTC should only be initialized once.

#### 4.49.3.2 sensor_port_info ports[ ]

**Initial value:**

```
= {
    { GPIO_PIN(8, 10), GPIO_PIN(2,  2), GPIO_PIN(0,  2),
      GPIO_PIN(0, 15), GPIO_PIN(8, 11), 0x6, 0x5 },
    { GPIO_PIN(8, 12), GPIO_PIN(8, 15), GPIO_PIN(0, 14),
      GPIO_PIN(0, 13), GPIO_PIN(8, 14), 0x8, 0x7 },
    { GPIO_PIN(8,  9), GPIO_PIN(7, 11), GPIO_PIN(0, 12),
      GPIO_PIN(1, 14), GPIO_PIN(7,  9), 0xA, 0x9 },
    { GPIO_PIN(6,  4), GPIO_PIN(7,  8), GPIO_PIN(0,  1),
      GPIO_PIN(1, 15), GPIO_PIN(7, 10), 0xC, 0xB },
}
```

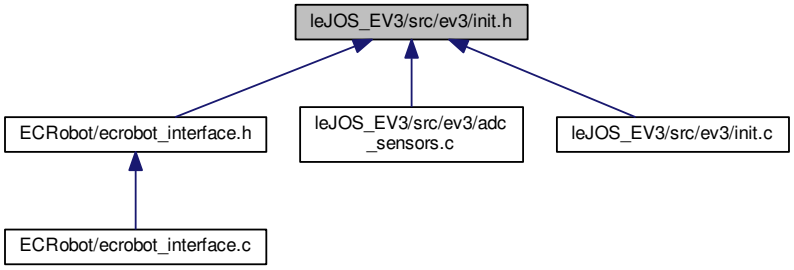
Array storing information about the 4 sensor ports of the EV3.

A internally used attribute. Needed for port access.

## 4.50 leJOS_EV3/src/ev3/init.h File Reference

This header provides the interface to initialize the leJOS EV3 drivers.

This graph shows which files directly or indirectly include this file:




## Data Structures

- struct sensor_port_info

  *This struct combines information about a sensor port of the EV3.*

## Typedefs

- typedef struct sensor_port_info sensor_port_info

  *This struct combines information about a sensor port of the EV3.*

## Functions

- void leJOS_init (void)

  *Initialize the leJOS driver and all required submodules.*

### 4.50.1 Detailed Description

This header provides the interface to initialize the leJOS EV3 drivers.

**Author**

Tobias Schießl, ev3ninja

### 4.50.2 Typedef Documentation

#### 4.50.2.1 typedef struct sensor_port_info sensor_port_info

This struct combines information about a sensor port of the EV3.

It is based on code taken from the ev3ninja project. Each sensor port is described by 4 GPIO pins which connect to the sensor, 1 buffer GPIO pin and 2 ADC channels.

### 4.50.3 Function Documentation

#### 4.50.3.1 void leJOS_init ( void )

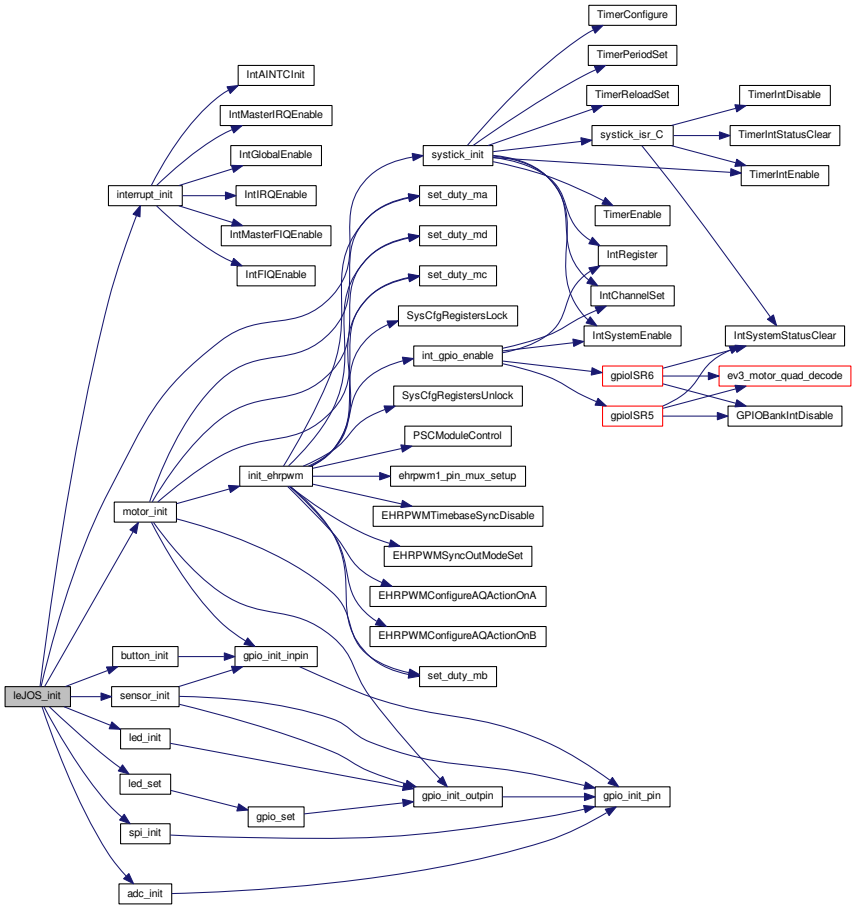
Initialize the leJOS driver and all required submodules.

Initialization contains: AINTC, systick, sensor ports, motor ports, SPI controller, ADC, hardware buttons, LEDs

**Returns**

none

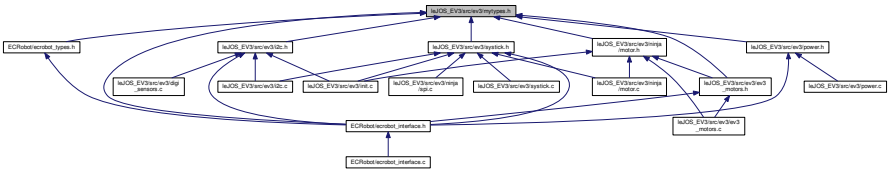
Here is the call graph for this function:




## 4.51 leJOS_EV3/src/ev3/mytypes.h File Reference

This header defines the data types used in the leJOS driver.

This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef unsigned char U8

  *An unsigned 8 bit (1 byte) data type.*
- typedef signed char S8

  *An signed 8 bit (1 byte) data type.*
- typedef unsigned short U16

  *An unsigned 16 bit (2 byte) data type.*
- typedef signed short S16

  *An signed 16 bit (2 byte) data type.*
- typedef unsigned long U32

  *An unsigned 32 bit (4 byte) data type.*
- typedef signed long S32

  *An signed 32 bit (4 byte) data type.*

### 4.51.1 Detailed Description

This header defines the data types used in the leJOS driver.

**Author**

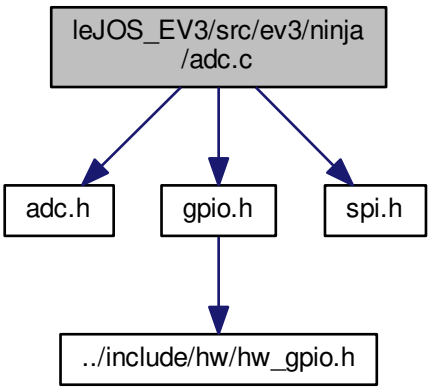
Tobias Schießl, NXT leJOS

## 4.52 leJOS_EV3/src/ev3/ninja/adc.c File Reference

This file contains function definitions to talk to the ADC.

```
#include "adc.h"
#include "gpio.h"
#include "spi.h"
```

Include dependency graph for adc.c:



## Functions

- unsigned short adc_get (unsigned short channel)

  *Get the value currently meassured by the ADC for the specified channel.*

- void adc_init (void)

  *Initialize the ADC by providing it with power.*

### 4.52.1 Detailed Description

This file contains function definitions to talk to the ADC.

**Author**

ev3ninja

### 4.52.2 Function Documentation

#### 4.52.2.1 unsigned short adc_get ( unsigned short *channel* )

Get the value currently meassured by the ADC for the specified channel.

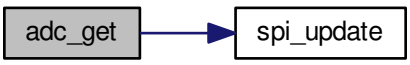
**Parameters**

| | |
|---|---|
| *channel* | - The channel to meassure (ranging from 0 to 15) |

**Returns**

The value currently meassured by the ADC (ranging from 0 to 4095)

Here is the call graph for this function:



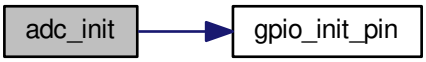

#### 4.52.2.2 void adc_init ( void )

Initialize the ADC by providing it with power.

**Returns**

none

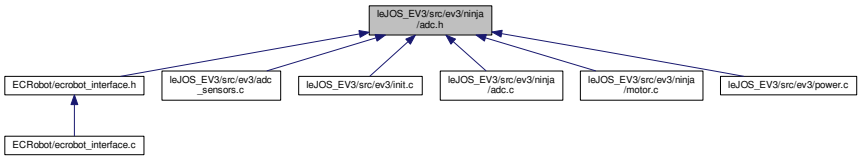
Here is the call graph for this function:




## 4.53 leJOS_EV3/src/ev3/ninja/adc.h File Reference

This header contains function declarations to talk to the ADC.

This graph shows which files directly or indirectly include this file:




### Functions

- unsigned short adc_get (unsigned short channel)

  *Get the value currently meassured by the ADC for the specified channel.*
- void adc_init (void)

  *Initialize the ADC by providing it with power.*

### 4.53.1 Detailed Description

This header contains function declarations to talk to the ADC.

**Author**

ev3ninja

### 4.53.2 Function Documentation

#### 4.53.2.1 unsigned short adc_get ( unsigned short *channel* )

Get the value currently meassured by the ADC for the specified channel.

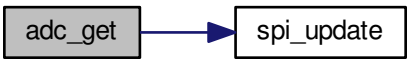
**Parameters**

| | |
|---|---|
| *channel* | - The channel to meassure (ranging from 0 to 15) |

**Returns**

The value currently meassured by the ADC (ranging from 0 to 4095)

Here is the call graph for this function:



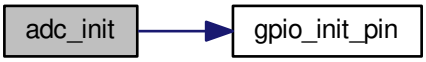

#### 4.53.2.2 void adc_init ( void )

Initialize the ADC by providing it with power.

**Returns**

none

Here is the call graph for this function:

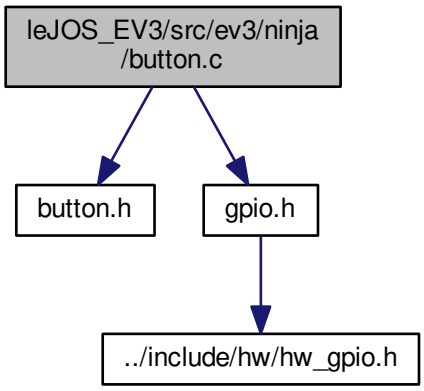



## 4.54 leJOS_EV3/src/ev3/ninja/button.c File Reference

This file contains function definitions to use the hardware buttons of the EV3.

```
#include "button.h"
#include "gpio.h"
```

Include dependency graph for button.c:




## Functions

- button_state button_get_state (button_id button)

  *Check if the given button is pressed.*
- button_id button_get_pressed (void)

  *Check if any button is pressed.*
- void button_init (void)

  *Initialize the buttons be configuring the required GPIO pins as input pins.*

## Variables

- button_info buttons [ ]

  *Array storing the required GPIO pins to read the state of the buttons.*

### 4.54.1 Detailed Description

This file contains function definitions to use the hardware buttons of the EV3.

**Author**

Tobias Schießl, ev3ninja

### 4.54.2 Function Documentation

#### 4.54.2.1 button_id button_get_pressed ( void )

Check if any button is pressed.

This function will check if any button is pressed. As soon as one button that is pressed is found, it will be returned and the remaining buttons will not be checked. The buttons are checked in the order they are defined in the enumeration button_id, i.e. LEFT, RIGHT, TOP, BOTTOM, CENTER, BACK.

**Returns**

The ID of the button that was pressed (ranging from 0x00 to 0x05) or BUTTON_NONE (0xFF) if none was pressed

Here is the call graph for this function:


#### 4.54.2.2 button_state button_get_state ( button_id *button* )

Check if the given button is pressed.

**Parameters**

| | |
|---|---|
| *button* | - The ID of the button to check (ranging from 0x00 to 0x05) |

**Returns**

The state of the button (BUTTON_DOWN = 0x01 if the button is pressed, BUTTON_UP = 0x00 otherwise)

Here is the call graph for this function:

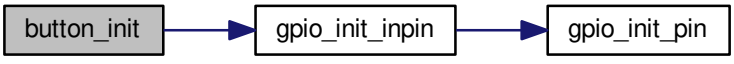

#### 4.54.2.3 void button_init ( void )

Initialize the buttons be configuring the required GPIO pins as input pins.

**Returns**

none

Here is the call graph for this function:



### 4.54.3 Variable Documentation

#### 4.54.3.1 button_info buttons[ ]

**Initial value:**

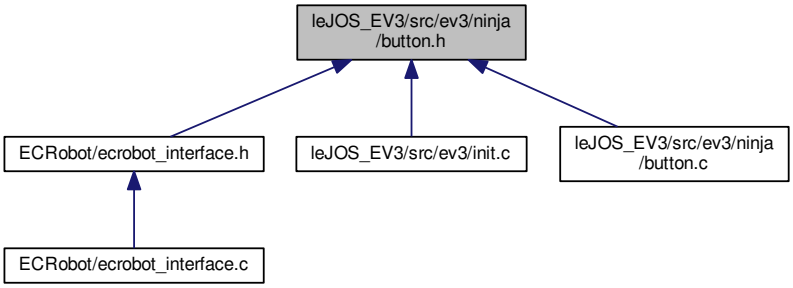
```
= {
    { GPIO_PIN(6,  6) },
    { GPIO_PIN(7, 12) },
    { GPIO_PIN(7, 15) },
    { GPIO_PIN(7, 14) },
    { GPIO_PIN(1, 13) },
    { GPIO_PIN(6, 10) }
}
```

Array storing the required GPIO pins to read the state of the buttons.

## 4.55 leJOS_EV3/src/ev3/ninja/button.h File Reference

This header contains function declarations to use the hardware buttons of the EV3 as well as enumerations to represent the buttons and their states.

This graph shows which files directly or indirectly include this file:




### Data Structures

- struct button_info

  *This struct contains the information required to read a buttons state.*

### Typedefs

- typedef struct button_info button_info

  *This struct contains the information required to read a buttons state.*
- typedef enum button_id button_id

  *This enumeration maps the buttons to unique IDs.*
- typedef enum button_state button_state

  *This enumeration represents a button's state (pressed = DOWN or not pressed = UP)*

### Enumerations

- enum button_id {
  BUTTON_LEFT = 0x00, BUTTON_RIGHT = 0x01, BUTTON_TOP = 0x02, BUTTON_BOTTOM = 0x03,
  BUTTON_CENTER = 0x04, BUTTON_BACK = 0x05, BUTTON_NONE = 0xFF }

  *This enumeration maps the buttons to unique IDs.*
- enum button_state { BUTTON_UP = 0x00, BUTTON_DOWN = 0x01 }

  *This enumeration represents a button's state (pressed = DOWN or not pressed = UP)*

### Functions

- button_state button_get_state (button_id button)

  *Check if the given button is pressed.*
- button_id button_get_pressed (void)

  *Check if any button is pressed.*
- void button_init (void)

  *Initialize the buttons be configuring the required GPIO pins as input pins.*

## 4.55.1 Detailed Description

This header contains function declarations to use the hardware buttons of the EV3 as well as enumerations to represent the buttons and their states.

**Author**

Tobias Schießl, ev3ninja

## 4.55.2 Typedef Documentation

### 4.55.2.1 typedef struct button_info button_info

This struct contains the information required to read a buttons state.

Every buttons is red by reading the signal on 1 GPIO pin which is configured as input.

## 4.55.3 Enumeration Type Documentation

### 4.55.3.1 enum button_id

This enumeration maps the buttons to unique IDs.

**Enumerator**

***BUTTON_LEFT*** The left button of the EV3.

***BUTTON_RIGHT*** The right button of the EV3.

***BUTTON_TOP*** The top button of the EV3.

***BUTTON_BOTTOM*** The bottom button of the EV3.

***BUTTON_CENTER*** The center button of the EV3.

***BUTTON_BACK*** The upper left button of the EV3.

***BUTTON_NONE*** ID representing no button.

#### 4.55.3.2 enum button_state

This enumeration represents a button's state (pressed = DOWN or not pressed = UP)

**Enumerator**

***BUTTON_UP*** State representing that the button was not pressed.

***BUTTON_DOWN*** State representing that the button was pressed.

### 4.55.4 Function Documentation

#### 4.55.4.1 button_id button_get_pressed ( void )

Check if any button is pressed.

This function will check if any button is pressed. As soon as one button that is pressed is found, it will be returned and the remaining buttons will not be checked. The buttons are checked in the order they are defined in the enumeration button_id, i.e. LEFT, RIGHT, TOP, BOTTOM, CENTER, BACK.

**Returns**

The ID of the button that was pressed (ranging from 0x00 to 0x05) or BUTTON_NONE (0xFF) if none was pressed

Here is the call graph for this function:


#### 4.55.4.2 button_state button_get_state ( button_id *button* )

Check if the given button is pressed.

**Parameters**

| | |
|---|---|
| *button* | - The ID of the button to check (ranging from 0x00 to 0x05) |

**Returns**

The state of the button (BUTTON_DOWN = 0x01 if the button is pressed, BUTTON_UP = 0x00 otherwise)

Here is the call graph for this function:

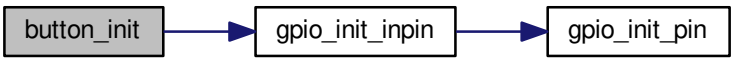
#### 4.55.4.3 void button_init ( void )

Initialize the buttons be configuring the required GPIO pins as input pins.

**Returns**

none

Here is the call graph for this function:

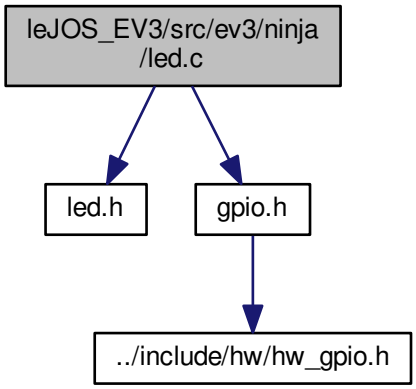



## 4.56 leJOS_EV3/src/ev3/ninja/led.c File Reference

This contains function definitions required to interact with the LEDs of the EV3.

```
#include "led.h"
#include "gpio.h"
```

Include dependency graph for led.c:




### Functions

- void led_set (led_id led, led_color color)

  *Set the color of the given LED.*
- void led_init (void)

  *Initialize the GPIO pins required to manipulate the LEDs.*

## Variables

- led_info leds [ ]
  *This array stores information about the 2 LEDs (left and right)*

### 4.56.1 Detailed Description

This contains function definitions required to interact with the LEDs of the EV3.

**Author**

ev3ninja

### 4.56.2 Function Documentation

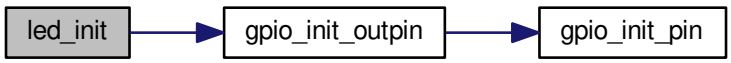
#### 4.56.2.1 void led_init ( void )

Initialize the GPIO pins required to manipulate the LEDs.

**Returns**

none

Here is the call graph for this function:




#### 4.56.2.2 void led_set ( led_id *led*, led_color *color* )

Set the color of the given LED.

**Parameters**

| | |
|---|---|
| *led* | - The ID of the LED to control |
| *color* | - The color to set |

**Returns**

none

Here is the call graph for this function:

### 4.56.3 Variable Documentation

#### 4.56.3.1 led_info leds[ ]

**Initial value:**

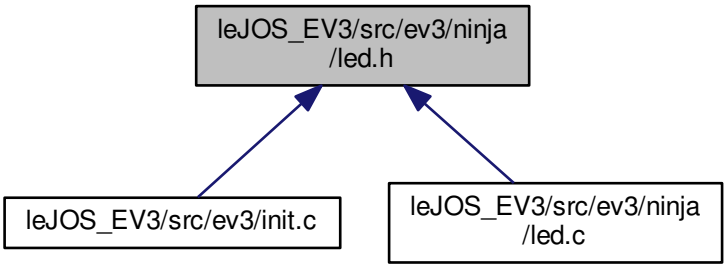
```
= {
    { GPIO_PIN(6, 13), GPIO_PIN(6,  7) },
    { GPIO_PIN(6, 12), GPIO_PIN(6, 14) }
}
```

This array stores information about the 2 LEDs (left and right)

## 4.57 leJOS_EV3/src/ev3/ninja/led.h File Reference

This header declares function required to interact with the LEDs of the EV3 as well as enumerations therefore.

This graph shows which files directly or indirectly include this file:




### Data Structures

- struct led_info

  *This struct stores information about one LED which is controlled by 2 GPIO pins.*

### Typedefs

- typedef enum led_id led_id

  *Enumeration to represent the ID of the LEDs.*
- typedef enum led_color led_color

  *Enumeration to represent the color which should be set for an LED.*
- typedef struct led_info led_info

  *This struct stores information about one LED which is controlled by 2 GPIO pins.*

### Enumerations

- enum led_id { LED_LEFT = 0x01, LED_RIGHT = 0x02, LED_BOTH = LED_LEFT | LED_RIGHT }

  *Enumeration to represent the ID of the LEDs.*

- enum led_color { LED_BLACK = 0x00, LED_RED = 0x01, LED_GREEN = 0x02, LED_ORANGE = LED_RED | LED_GREEN }

  *Enumeration to represent the color which should be set for an LED.*

## Functions

- void led_set (led_id led, led_color color)

  *Set the color of the given LED.*
- void led_init (void)

  *Initialize the GPIO pins required to manipulate the LEDs.*

### 4.57.1 Detailed Description

This header declares function required to interact with the LEDs of the EV3 as well as enumerations therefore.

**Author**

ev3ninja

### 4.57.2 Typedef Documentation

#### 4.57.2.1 typedef enum led_id led_id

Enumeration to represent the ID of the LEDs.

Every LED in fact consists of two LEDs, a red ond and a green one.

#### 4.57.2.2 typedef struct led_info led_info

This struct stores information about one LED which is controlled by 2 GPIO pins.

One GPIO pin controlls the red LED and one pin the green LED.

### 4.57.3 Enumeration Type Documentation

#### 4.57.3.1 enum led_color

Enumeration to represent the color which should be set for an LED.

**Enumerator**

***LED_BLACK*** Black color (which equals off)

***LED_RED*** Red color (which equals enabling only one GPIO pin)

***LED_GREEN*** Green color (which equals enabling only one GPIO pin)

***LED_ORANGE*** Orange color (which equals enabling both GPIO pins)

#### 4.57.3.2 enum led_id

Enumeration to represent the ID of the LEDs.

Every LED in fact consists of two LEDs, a red ond and a green one.

**Enumerator**

***LED_LEFT*** The left LEDs (red and green) of the EV3.

***LED_RIGHT*** The right LEDs (red and green) of the EV3.

***LED_BOTH*** Both LEDs of the EV3.

### 4.57.4 Function Documentation

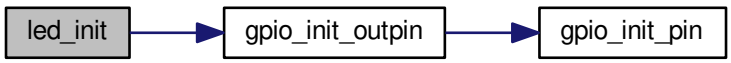
#### 4.57.4.1 void led_init ( void )

Initialize the GPIO pins required to manipulate the LEDs.

**Returns**

none

Here is the call graph for this function:




#### 4.57.4.2 void led_set ( led_id *led*, led_color *color* )

Set the color of the given LED.

**Parameters**

| | |
|---|---|
| *led* | - The ID of the LED to control |
| *color* | - The color to set |

**Returns**

none

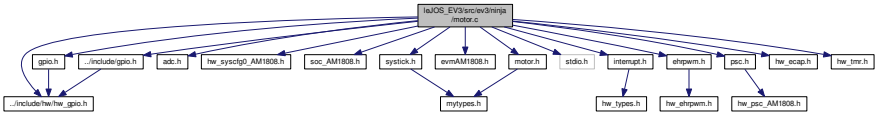
Here is the call graph for this function:


## 4.58 leJOS_EV3/src/ev3/ninja/motor.c File Reference

Driver implementation to control LEGO servo motors. This file contains function definitions to use motors with EV3.

```
#include "motor.h"
#include "gpio.h"
#include "adc.h"
#include "hw_syscfg0_AM1808.h"
#include "soc_AM1808.h"
#include "interrupt.h"
#include "evmAM1808.h"
#include "ehrpwm.h"
#include "stdio.h"
#include "psc.h"
#include "hw_gpio.h"
#include "../include/gpio.h"
#include "systick.h"
#include "hw_ecap.h"
#include "hw_tmr.h"
```

Include dependency graph for motor.c:




## Macros

- #define PINMUX5_EPWM1A_ENABLE (SYSCFG_PINMUX5_PINMUX5_3_0_EPWM1A << SYSCFG_P↩ INMUX5_PINMUX5_3_0_SHIFT)

  *Pin Multiplexing bit mask to select EPWM1A pin.*
- #define PINMUX5_EPWM1B_ENABLE (SYSCFG_PINMUX5_PINMUX5_7_4_EPWM1B << SYSCFG_P↩ INMUX5_PINMUX5_7_4_SHIFT)

  *Pin Multiplexing bit mask to select EPWM1B pin.*
- #define PINMUX2_ECAP0_ENABLE (SYSCFG_PINMUX2_PINMUX2_31_28_ECAP0 << SYSCFG_PIN↩ MUX2_PINMUX2_31_28_SHIFT)

  *Pin Multiplexing bit mask to select ECAP0 pin.*
- #define PINMUX1_ECAP1_ENABLE (SYSCFG_PINMUX1_PINMUX1_31_28_ECAP1 << SYSCFG_PIN↩ MUX1_PINMUX1_31_28_SHIFT)

  *Multiplexing bit mask to select ECAP1 pin.*
- #define MAX_PWM_CNT (10000)

  *Maximal nuber that pwm counts to is 10000. This value was taken from ev3sources.*

## Functions

- void ehrpwm1_pin_mux_setup (void)
  - *This function does appropriate Pin multiplexing to enable the use of pwm1, ecap0 and ecap1 related pins on the board.*
- void int_gpio_enable (void)

  *Enable GPIO interrupts for bank 5 and 6.*
- void init_ehrpwm (void)
  - *Initialize EHRPWM.*
- void set_duty_ma (U32 duty)
  - *Load Compare A value*
- void set_duty_mb (U32 duty)
  - *Load Compare B value*

- void set_duty_mc (U32 duty)
  - *Load Compare C value*
- void set_duty_md (U32 duty)
  - *Load Compare D value*
- void set_power (motor_port_id port, S32 power)
  - *Set power percent applied to motors. The calculation are appropriated to nxt motors*
- motor_type_id get_motor_type (motor_port_id Port)

  *This function is not used. For future use. adc ch.14 -> A adc ch.13 -> B adc ch.0 -> C adc ch.1 -> D NXT_SERVO_ID (124) EV3_MEDIUM_MOTOR_ID (288) EV3_LARGE_MOTOR_ID (3692)*
- int ev3_get_count (int motor_port_id)

  *Get motor revolution count in degree.*
- void ev3_set_count (int motor_port_id, int count)
  - *Set motor revolution count in degree*
- void ev3_motor_command (U32 motor_port_id, int cmd, int target_count, int speed_percent)

  *Set motor target revolution count, brake mode and speed percent. After reaching the target count the given motor stops.*
- void set_brake_mode (int motor_port_id, int brake_mode)

  *Set brake mode. Brake - stop immediately, float - soft stop.*
- void motor_set_state (motor_port_id port, motor_state state)

  *Set the state of an attached motor.*
- unsigned int get_tacho_dir (motor_port_id port)

  *Get GPIO value of pin 6 (DIR). Is used to calculate motor revolution.*
- unsigned int get_tacho_int (motor_port_id port)

  *Get GPIO value of pin 5 (INT). Is used to calculate motor revolution.*
- void motor_init (void)

  *Initialize the GPIO pins and the pwm modules necessary for motor moving functions.*
- void gpioISR5 (void)

  *Interrupt service routine for Pin Bank 5 (ports A, B and C). Calls ev3_motor_quad_decode() if one of the 3 GPIO pins caused this interrupt. Is used to calculate motor revolution in degrees.*
- void gpioISR6 (void)

  *Interrupt service routine for Pin Bank 6. Calls ev3_motor_quad_decode() to actualize current wheel revolution count of motor on port D.*
- void ev3_motor_quad_decode (int motor_port_id)

  *Calculate actual revolution count of given motor port.*

## Variables

- volatile motor_port_info * **addrMotorPorts** = ports
- motor_data_struct motor_data [4]

  *Array storing the information about motor speed, current and target revolution counts and brake mode.*
- unsigned int masks [3] = {0x08000000, 0x01000000, 0x20000000}

  *Array storing the required GPIO pin masks for motors on ports A, B and C.*

### 4.58.1 Detailed Description

Driver implementation to control LEGO servo motors. This file contains function definitions to use motors with EV3.

LEGO EV3 can be plugged with 4 motors on ports A, B, C and D of the brick. The Enhanced High Resolution Pulse Width Modulator(EHRPWM) of TexasInstruments Sitara AM1808 SoC manages motors speed. There are two ways to stop motors: brake (immediate stop) and float (soft stop). You can get and set rotation angle of motors in degrees anytime. Additionaly you can set a target degree for some motor to reach and stop.

The identification of motor types (large, medium and nxt) was not implemented. So all speed calculations are configured for nxt motors.

**Author**

Bektur Marat uulu and Bektur Toktosunov

### 4.58.2 Function Documentation

#### 4.58.2.1 void ehrpwm1_pin_mux_setup ( void )

- This function does appropriate Pin multiplexing to enable the use of pwm1, ecap0 and ecap1 related pins on the board.

**Returns**

None.

#### 4.58.2.2 int ev3_get_count ( int *motor_port_id* )

Get motor revolution count in degree.

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

revolution in degree

#### 4.58.2.3 void ev3_motor_command ( U32 *motor_port_id,* int *cmd,* int *target_count,* int *speed_percent* )

Set motor target revolution count, brake mode and speed percent. After reaching the target count the given motor stops.

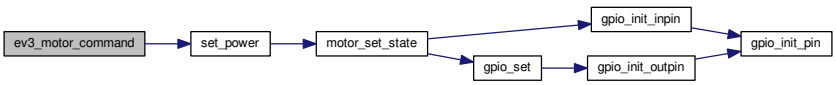
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |
| *cmd* | - brake mode. true - brake, false - float |
| *target_count* | - Target revolution count to reach in degree |
| *speed_percent* | - Speed percent to rotate |

**Returns**

none

Here is the call graph for this function:




#### 4.58.2.4 void ev3_motor_quad_decode ( int *motor_port_id* )

Calculate actual revolution count of given motor port.

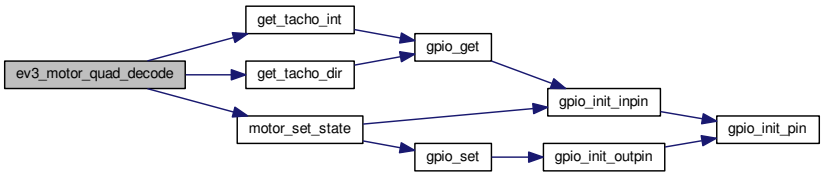
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

none

Here is the call graph for this function:




#### 4.58.2.5 void ev3_set_count ( int *motor_port_id,* int *count* )

- Set motor revolution count in degree

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |
| *count* | - Motor revolution count to set |

**Returns**

none

#### 4.58.2.6 unsigned int get_tacho_dir ( motor_port_id *port* )

Get GPIO value of pin 6 (DIR). Is used to calculate motor revolution.

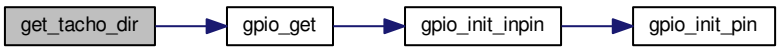
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

Pin 6 value of a given port

Here is the call graph for this function:



#### 4.58.2.7 unsigned int get_tacho_int ( motor_port_id *port* )

Get GPIO value of pin 5 (INT). Is used to calculate motor revolution.

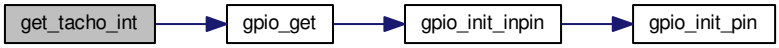
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

Pin 5 value of a given port

Here is the call graph for this function:



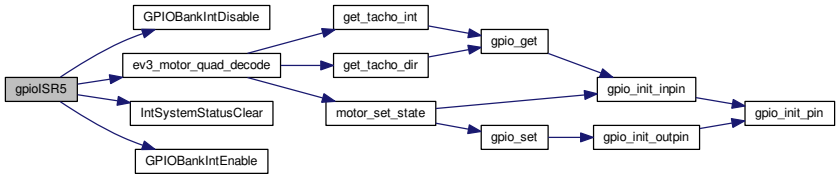

#### 4.58.2.8 void gpioISR5 ( void )

Interrupt service routine for Pin Bank 5 (ports A, B and C). Calls ev3_motor_quad_decode() if one of the 3 GPIO pins caused this interrupt. Is used to calculate motor revolution in degrees.

**Returns**

none

Here is the call graph for this function:



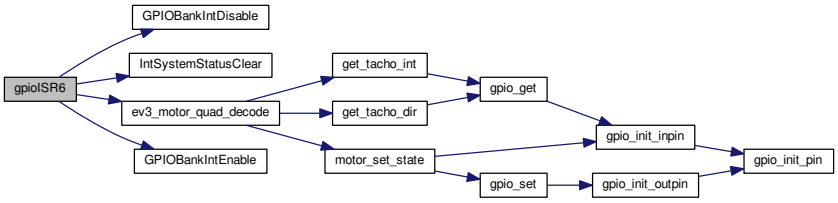

#### 4.58.2.9 void gpioISR6 ( void )

Interrupt service routine for Pin Bank 6. Calls ev3_motor_quad_decode() to actualize current wheel revolution count of motor on port D.

**Returns**

none

Here is the call graph for this function:




#### 4.58.2.10 void init_ehrpwm ( void )

- Initialize EHRPWM.

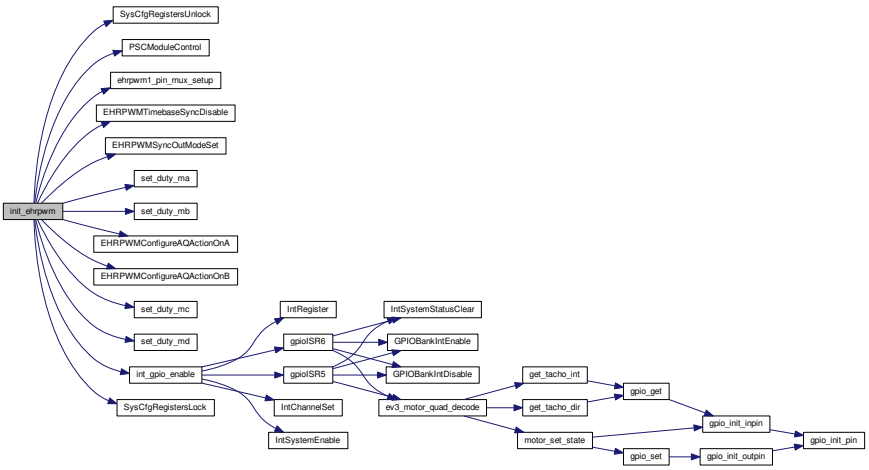
Enable EHRPWM and ECAP modules. PWM counts from 0 to counter period (10000). Set compare values of pwm and ecap modules to 0. A duty cicle of a motor begins when PWM counter reaches a given compare value and ends by reaching the counter period value (10000).

**Returns**

none

for Motor A and Motor B

Here is the call graph for this function:



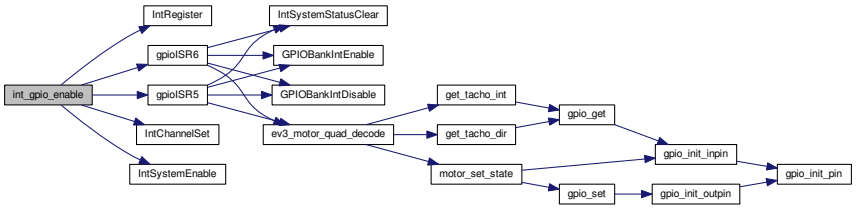

#### 4.58.2.11 void int_gpio_enable ( void )

Enable GPIO interrupts for bank 5 and 6.

**Returns**

none

Here is the call graph for this function:



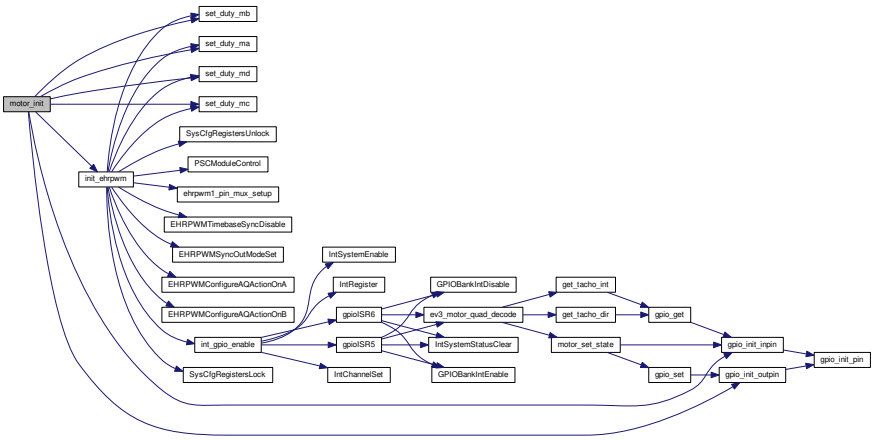

#### 4.58.2.12 void motor_init ( void )

Initialize the GPIO pins and the pwm modules necessary for motor moving functions.

**Returns**

none

Here is the call graph for this function:




#### 4.58.2.13 void motor_set_state ( motor_port_id *port*, motor_state *state* )

Set the state of an attached motor.

**Parameters**

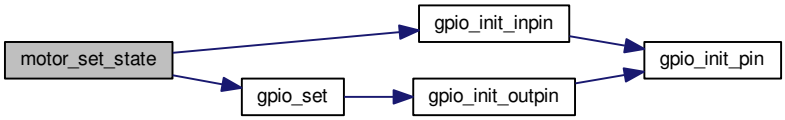
| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_POR↩ T_D) |

| | |
|---|---|
| *state* | - one of MOTOR_FORWARD, MOTOR_BACKWARD, MOTOR_OFF |

**Returns**

none

Here is the call graph for this function:




#### 4.58.2.14 void set_brake_mode ( int *motor_port_id,* int *brake_mode* )

Set brake mode. Brake - stop immediately, float - soft stop.

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_POR↩ T_D) - Brake mode. True - brake, false - float |

**Returns**

none

#### 4.58.2.15 void set_duty_ma ( U32 *duty* )

- Load Compare A value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value epwm1A (Port B) |

**Returns**

none

#### 4.58.2.16 void set_duty_mb ( U32 *duty* )

- Load Compare B value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value of epwm1B (Port A) |

**Returns**

none

#### 4.58.2.17 void set_duty_mc ( U32 *duty* )

- Load Compare C value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value of ecap1 (Port C) |

**Returns**

none

#### 4.58.2.18 void set_duty_md ( U32 *duty* )

- Load Compare D value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value of ecap0 (Port D) |

**Returns**

none

#### 4.58.2.19 void set_power ( motor_port_id *port*, S32 *power* )

- Set power percent applied to motors. The calculation are appropriated to nxt motors

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_POR↩ T_D) |
| *power* | - Power percent from -100 to 100. Rotate backward if a negative number is given. |

**Returns**

none

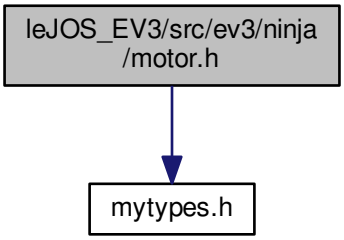
Here is the call graph for this function:

## 4.59 leJOS_EV3/src/ev3/ninja/motor.h File Reference

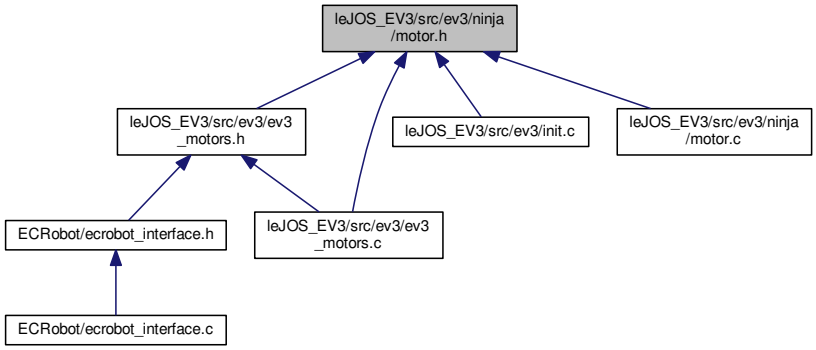
A header file for motor driver. This file contains function declarations to use servo motors as well as enumerations and structs to represent motors and their states.

```
#include "mytypes.h"
```

Include dependency graph for motor.h:




This graph shows which files directly or indirectly include this file:




### Data Structures

- struct motor_port_info

  *Contains the gpio pins of output and input pins.*
- struct motor_data_struct

  *Contains the information about wheel revolution, given speed and brake mode of motors.*

### Macros

- #define NO_OF_OUTPUT_PORTS 4

  *Number of output ports of servo motors.*

## Typedefs

- typedef struct motor_port_info **motor_port_info**
- typedef struct motor_data_struct **motor_data_struct**
- typedef enum motor_port_id **motor_port_id**
- typedef enum motor_type_id **motor_type_id**
- typedef enum motor_state **motor_state**

## Enumerations

- enum motor_port_id { **MOTOR_PORT_A** = 0x00, **MOTOR_PORT_B** = 0x01, **MOTOR_PORT_C** = 0x02, **MOTOR_PORT_D** = 0x03 }

  *Enumeration of motor ports.*
- enum motor_type_id { **EV3_MEDIUM_MOTOR**, **NXT_SERVO_MOTOR**, **UNKNOWN_MOTOR_TYPE** }

  *Enumeration of motor types.*
- enum motor_state { **MOTOR_BACKWARD** = 0x00, **MOTOR_OFF** = 0x01, **MOTOR_FORWARD** = 0x02 }

  *Enumeration of motor states.*

## Functions

- void motor_set_state (motor_port_id port, motor_state state)

  *Set the state of an attached motor.*
- void set_duty_ma (U32 duty)
  - *Load Compare A value*
- void set_duty_mb (U32 duty)
  - *Load Compare B value*
- void set_duty_mc (U32 duty)
  - *Load Compare C value*
- void set_duty_md (U32 duty)
  - *Load Compare D value*
- unsigned int get_tacho_dir (motor_port_id Port)

  *Get GPIO value of pin 6 (DIR). Is used to calculate motor revolution.*
- unsigned int get_tacho_int (motor_port_id Port)

  *Get GPIO value of pin 5 (INT). Is used to calculate motor revolution.*
- motor_type_id get_motor_type (motor_port_id Port)

  *This function is not used. For future use. adc ch.14 -> A adc ch.13 -> B adc ch.0 -> C adc ch.1 -> D NXT_SER↩ VO_ID (124) EV3_MEDIUM_MOTOR_ID (288) EV3_LARGE_MOTOR_ID (3692)*
- void set_power (motor_port_id Port, S32 Power)
  - *Set power percent applied to motors. The calculation are appropriated to nxt motors*
- void motor_init (void)

  *Initialize the GPIO pins and the pwm modules necessary for motor moving functions.*
- int ev3_get_count (int motor_port_id)

  *Get motor revolution count in degree.*
- void ev3_set_count (int motor_port_id, int count)
  - *Set motor revolution count in degree*
- void ev3_motor_command (U32 motor_port_id, int cmd, int target_count, int speed_percent)

  *Set motor target revolution count, brake mode and speed percent. After reaching the target count the given motor stops.*
- void ev3_motor_quad_decode (int motor_port_id)

  *Calculate actual revolution count of given motor port.*
- void set_brake_mode (int motor_port_id, int brake_mode)

  *Set brake mode. Brake - stop immediately, float - soft stop.*

### 4.59.1 Detailed Description

A header file for motor driver. This file contains function declarations to use servo motors as well as enumerations and structs to represent motors and their states.

**Author**

Bektur Marat uulu and Bektur Toktosunov

### 4.59.2 Function Documentation

#### 4.59.2.1 int ev3_get_count ( int *motor_port_id* )

Get motor revolution count in degree.

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

revolution in degree

#### 4.59.2.2 void ev3_motor_command ( U32 *motor_port_id,* int *cmd,* int *target_count,* int *speed_percent* )

Set motor target revolution count, brake mode and speed percent. After reaching the target count the given motor stops.

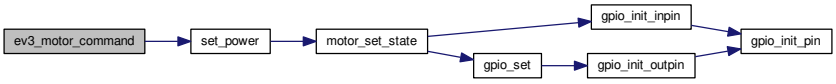
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |
| *cmd* | - brake mode. true - brake, false - float |
| *target_count* | - Target revolution count to reach in degree |
| *speed_percent* | - Speed percent to rotate |

**Returns**

none

Here is the call graph for this function:




#### 4.59.2.3 void ev3_motor_quad_decode ( int *motor_port_id* )

Calculate actual revolution count of given motor port.

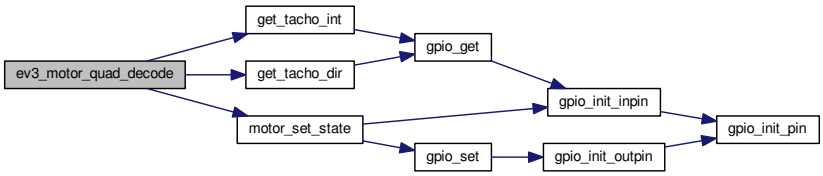
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

none

Here is the call graph for this function:




#### 4.59.2.4 void ev3_set_count ( int *motor_port_id,* int *count* )

- Set motor revolution count in degree

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |
| *count* | - Motor revolution count to set |

**Returns**

none

#### 4.59.2.5 unsigned int get_tacho_dir ( motor_port_id *port* )

Get GPIO value of pin 6 (DIR). Is used to calculate motor revolution.

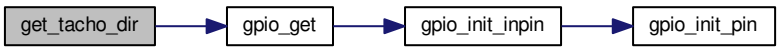
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

Pin 6 value of a given port

Here is the call graph for this function:



#### 4.59.2.6 unsigned int get_tacho_int ( motor_port_id *port* )

Get GPIO value of pin 5 (INT). Is used to calculate motor revolution.

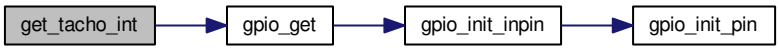
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |

**Returns**

Pin 5 value of a given port

Here is the call graph for this function:



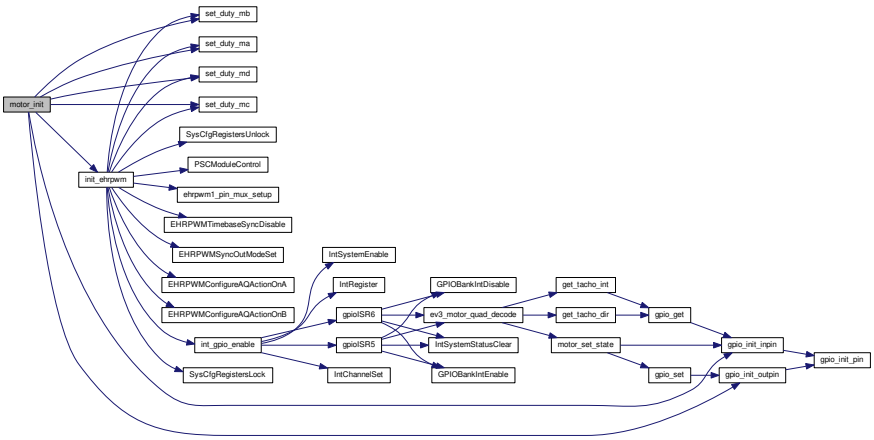

#### 4.59.2.7 void motor_init ( void )

Initialize the GPIO pins and the pwm modules necessary for motor moving functions.

**Returns**

none

Here is the call graph for this function:




#### 4.59.2.8 void motor_set_state ( motor_port_id *port,* motor_state *state* )

Set the state of an attached motor.

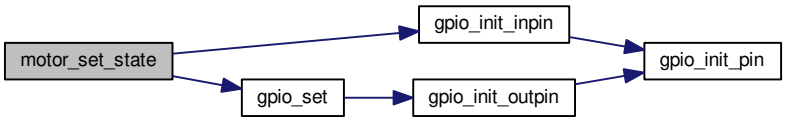
**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) |
| *state* | - one of MOTOR_FORWARD, MOTOR_BACKWARD, MOTOR_OFF |

**Returns**

none

Here is the call graph for this function:




#### 4.59.2.9 void set_brake_mode ( int *motor_port_id,* int *brake_mode* )

Set brake mode. Brake - stop immediately, float - soft stop.

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_PORT_D) - Brake mode. True - brake, false - float |

**Returns**

none

#### 4.59.2.10 void set_duty_ma ( U32 *duty* )

- Load Compare A value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value epwm1A (Port B) |

**Returns**

none

#### 4.59.2.11 void set_duty_mb ( U32 *duty* )

- Load Compare B value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value of epwm1B (Port A) |

**Returns**

none

#### 4.59.2.12 void set_duty_mc ( U32 *duty* )

- Load Compare C value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value of ecap1 (Port C) |

**Returns**

none

#### 4.59.2.13 void set_duty_md ( U32 *duty* )

- Load Compare D value

**Parameters**

| | |
|---|---|
| *duty* | - Compare value of ecap0 (Port D) |

**Returns**

none

#### 4.59.2.14 void set_power ( motor_port_id *port*, S32 *power* )

- Set power percent applied to motors. The calculation are appropriated to nxt motors

**Parameters**

| | |
|---|---|
| *motor_port_id* | - Motor port id (MOTOR_PORT_A, MOTOR_PORT_B, MOTOR_PORT_C, MOTOR_POR↩ T_D) |
| *power* | - Power percent from -100 to 100. Rotate backward if a negative number is given. |

**Returns**

none

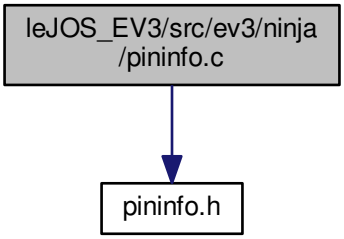
Here is the call graph for this function:

## 4.60 leJOS_EV3/src/ev3/ninja/pininfo.c File Reference

This file defines the array required to manipulate the PINMUX registers and set the functionality of GPIO pins.

```
#include "pininfo.h"
```

Include dependency graph for pininfo.c:




### Variables

- pin_info pininfo [ ]

  *The array containing the information to manipulate the PINMUX registers.*
- unsigned int pininfo_size = sizeof(pininfo) / sizeof(pininfo[0])

  *The size of the pininfo array (defined by the number of entries)*

### 4.60.1 Detailed Description

This file defines the array required to manipulate the PINMUX registers and set the functionality of GPIO pins.

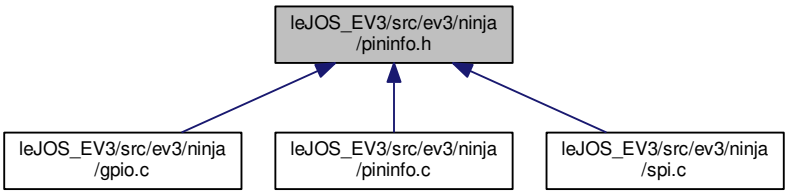
**Author**

ev3ninja

## 4.61 leJOS_EV3/src/ev3/ninja/pininfo.h File Reference

This header defines the struct that represents a GPIO pin in the ev3ninja driver and makes the corresponding variable in pininfo.c accessible by other files.

This graph shows which files directly or indirectly include this file:



### Data Structures

- struct pin_info

  *This struct represents 1 GPIO pin.*

### Typedefs

- typedef struct pin_info pin_info

  *This struct represents 1 GPIO pin.*

### Variables

- pin_info pininfo [ ]

  *The array containing the information to manipulate the PINMUX registers.*
- unsigned int pininfo_size

  *The size of the pininfo array (defined by the number of entries)*

#### 4.61.1 Detailed Description

This header defines the struct that represents a GPIO pin in the ev3ninja driver and makes the corresponding variable in pininfo.c accessible by other files.

**Author**

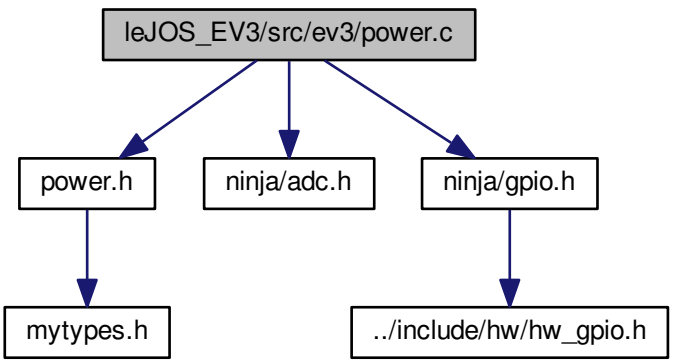
ev3ninja

## 4.62 leJOS_EV3/src/ev3/power.c File Reference

This header contains function definitions for the power management on the EV3.

```
#include "power.h"
#include "ninja/adc.h"
#include "ninja/gpio.h"
```

Include dependency graph for power.c:



## Functions

- void power_off (void)

  *Turn of the EV3.*

- U16 get_battery_voltage (void)

  *Get the voltage of the battery.*

- U16 get_battery_vurrent (void)

  *Get the current of the battery.*

### 4.62.1 Detailed Description

This header contains function definitions for the power management on the EV3.

**Author**

Tobias Schießl, Bektur Marat uulu, Bektur Toktosunov

### 4.62.2 Function Documentation

#### 4.62.2.1 U16 get_battery_voltage ( void )

Get the voltage of the battery.

**Returns**

The volatge of the battery

Here is the call graph for this function:

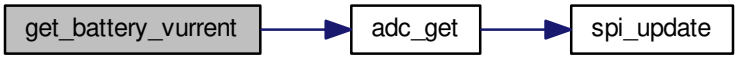

#### 4.62.2.2 U16 get_battery_vurrent ( void )

Get the current of the battery.

**Returns**

The current of the battery

Here is the call graph for this function:



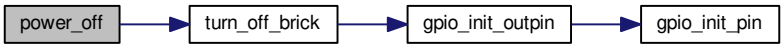

#### 4.62.2.3 void power_off ( void )

Turn of the EV3.

**Returns**

none

Here is the call graph for this function:

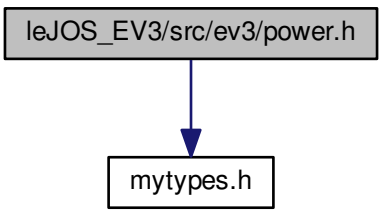



## 4.63 leJOS_EV3/src/ev3/power.h File Reference

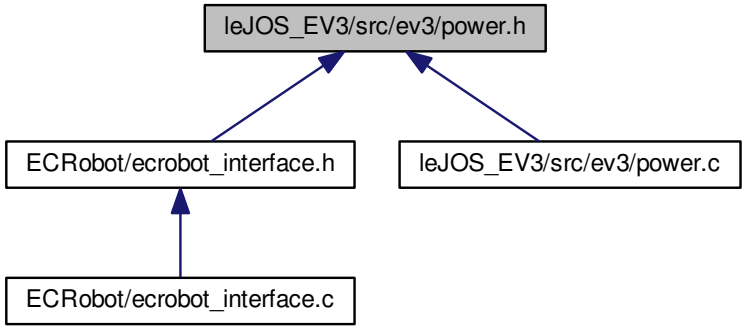
This header contains function declarations for the power management on the EV3.

```
#include "mytypes.h"
```

Include dependency graph for power.h:



This graph shows which files directly or indirectly include this file:




## Functions

- void power_off (void)

  *Turn of the EV3.*
- U16 get_battery_voltage (void)

  *Get the voltage of the battery.*
- U16 get_battery_vurrent (void)

  *Get the current of the battery.*

### 4.63.1 Detailed Description

This header contains function declarations for the power management on the EV3.

**Author**

Tobias Schießl

### 4.63.2 Function Documentation

#### 4.63.2.1 U16 get_battery_voltage ( void )

Get the voltage of the battery.

**Returns**

The volatge of the battery

Here is the call graph for this function:


#### 4.63.2.2 U16 get_battery_vurrent ( void )

Get the current of the battery.

**Returns**

The current of the battery

Here is the call graph for this function:

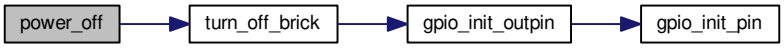

#### 4.63.2.3 void power_off ( void )

Turn of the EV3.

**Returns**

none

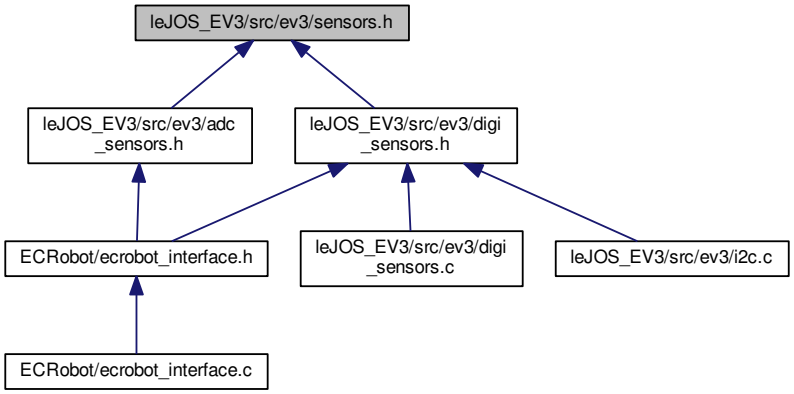
Here is the call graph for this function:



## 4.64 leJOS_EV3/src/ev3/sensors.h File Reference

This header contains macros for the sensor ports of the EV3.

This graph shows which files directly or indirectly include this file:




### Macros

- #define NXT_PORT_S1 0

  *NXT sensor port 1.*
- #define NXT_PORT_S2 1

  *NXT sensor port 2.*
- #define NXT_PORT_S3 2

  *NXT sensor port 3.*
- #define NXT_PORT_S4 3

  *NXT sensor port 4.*
- #define EV3_PORT_S1 NXT_PORT_S1

  *EV3 sensor port 1.*
- #define EV3_PORT_S2 NXT_PORT_S2

  *EV3 sensor port 2.*
- #define EV3_PORT_S3 NXT_PORT_S3

  *EV3 sensor port 3.*
- #define EV3_PORT_S4 NXT_PORT_S4

  *EV3 sensor port 4.*

### 4.64.1 Detailed Description

This header contains macros for the sensor ports of the EV3.

**Author**

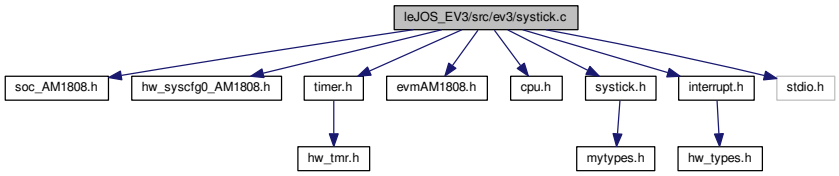
Tobias Schießl

## 4.65 leJOS_EV3/src/ev3/systick.c File Reference

This header contains function definitions required to manage the tick in milliseconds on the EV3.

```
#include "soc_AM1808.h"
#include "hw_syscfg0_AM1808.h"
#include "timer.h"
#include "evmAM1808.h"
#include "cpu.h"
#include "systick.h"
#include "interrupt.h"
#include "stdio.h"
```

Include dependency graph for systick.c:




### Macros

- #define TMR_PERIOD_LSB32 0x05CC

  *The compare value to set for the 16 least significant bits of the hardware timer.*
- #define TMR_PERIOD_MSB32 0x0

  *The compare value to set for the 16 most significant bits of the hardware timer.*

### Functions

- void systick_isr_C (void)

  *The systick interrupt service routine (ISR) which will be called every millisecond.*
- U32 systick_get_ms (void)

  *Get the current tick in milliseconds.*
- void systick_wait_ms (U32 ms)

  *Wait for the specified amount of time.*
- void systick_wait_ns (U32 ns)

  *Wait for the specified amount of time.*
- void systick_init (void)

  *Initialize the systick module, i.e. the hardware timer of the SoC.*
- void systick_suspend (void)

  *Disable the timer and therefore the systick.*
- void systick_resume (void)

  *Enable the timer and therefore the systick.*

### Variables

- volatile U32 systick_ms = 0

  *The current tick in milliseconds.*

### 4.65.1 Detailed Description

This header contains function definitions required to manage the tick in milliseconds on the EV3.

The tick is provided by a hardware timer of the AM1808 SoC which will trigger an interrupt every millisecond. The timer runs in 32 bit mode.

**Author**

Tobias Schießl

### 4.65.2 Macro Definition Documentation

#### 4.65.2.1 #define TMR_PERIOD_LSB32 0x05CC

The compare value to set for the 16 least significant bits of the hardware timer.

This is the value which causes the interrupt to be triggered every millisecond.

### 4.65.3 Function Documentation

#### 4.65.3.1 U32 systick_get_ms ( void )

Get the current tick in milliseconds.

**Returns**

The current tick in milliseconds

#### 4.65.3.2 void systick_init ( void )

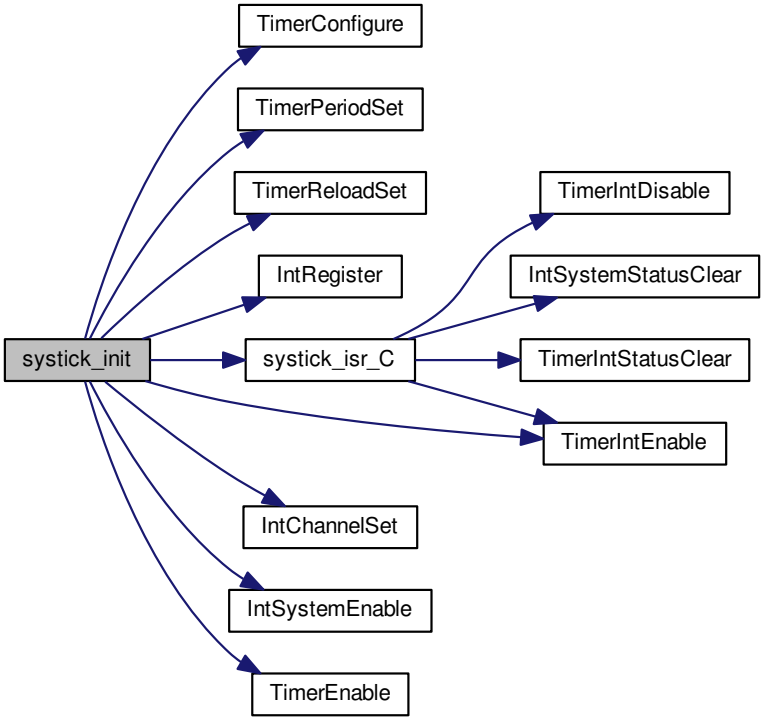
Initialize the systick module, i.e. the hardware timer of the SoC.

This function will register the corresponding ISR, enable the timer interrupt and configure interrupt channel 2 (normal interrupt) for the hardware timer.

**Returns**

none

Here is the call graph for this function:




#### 4.65.3.3 void systick_isr_C ( void )

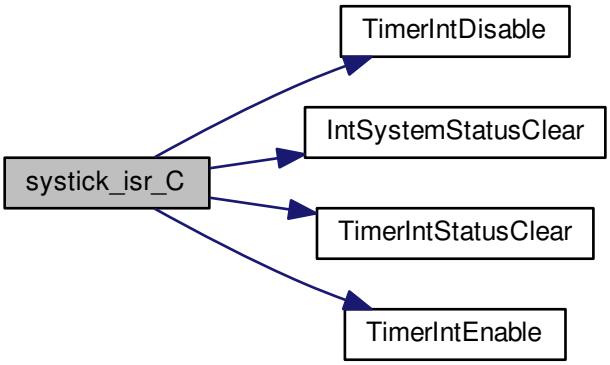
The systick interrupt service routine (ISR) which will be called every millisecond.

This ISR will increase the systick_ms variable and reset the interrupt flags.

**Returns**

none

Here is the call graph for this function:



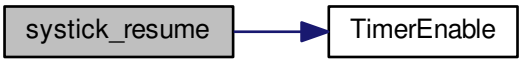

#### 4.65.3.4 void systick_resume ( void )

Enable the timer and therefore the systick.

**Returns**

none

Here is the call graph for this function:



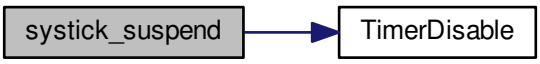

#### 4.65.3.5 void systick_suspend ( void )

Disable the timer and therefore the systick.

**Returns**

none

Here is the call graph for this function:




#### 4.65.3.6 void systick_wait_ms ( U32 *ms* )

Wait for the specified amount of time.

Waiting with this function is an active waiting which will block until the time has elapsed.

**Parameters**

| | |
|---|---|
| *ms* | - The time to wait in milliseconds |

**Returns**

none

#### 4.65.3.7 void systick_wait_ns ( U32 *ns* )

Wait for the specified amount of time.

Waiting with this function is an active waiting which will block until the time has elapsed.

**Parameters**

| | |
|---|---|
| *ns* | - The time to wait in nanoseconds |

**Returns**

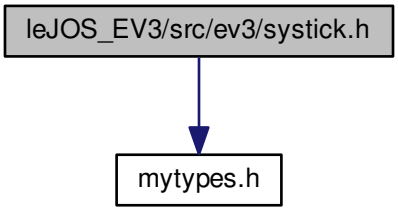
none

## 4.66 leJOS_EV3/src/ev3/systick.h File Reference

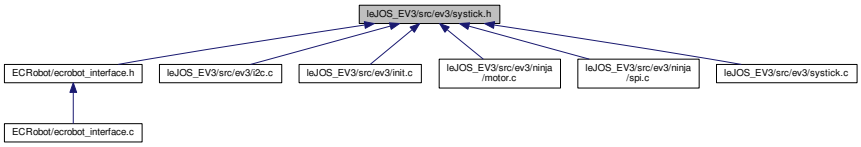
This header contains function declarations required to manage the tick in milliseconds on the EV3.

```
#include "mytypes.h"
```

Include dependency graph for systick.h:




This graph shows which files directly or indirectly include this file:




### Functions

- void systick_init (void)

  *Initialize the systick module, i.e. the hardware timer of the SoC.*
- U32 systick_get_ms (void)

  *Get the current tick in milliseconds.*
- void systick_wait_ms (U32 ms)

  *Wait for the specified amount of time.*
- void systick_wait_ns (U32 n)

  *Wait for the specified amount of time.*
- void systick_suspend (void)

  *Disable the timer and therefore the systick.*
- void systick_resume (void)

  *Enable the timer and therefore the systick.*

### 4.66.1 Detailed Description

This header contains function declarations required to manage the tick in milliseconds on the EV3.

The tick is provided by a hardware timer of the AM1808 SoC which will trigger an interrupt every millisecond.

**Author**

Tobias Schießl

### 4.66.2 Function Documentation

#### 4.66.2.1 U32 systick_get_ms ( void )

Get the current tick in milliseconds.

**Returns**

The current tick in milliseconds

#### 4.66.2.2 void systick_init ( void )

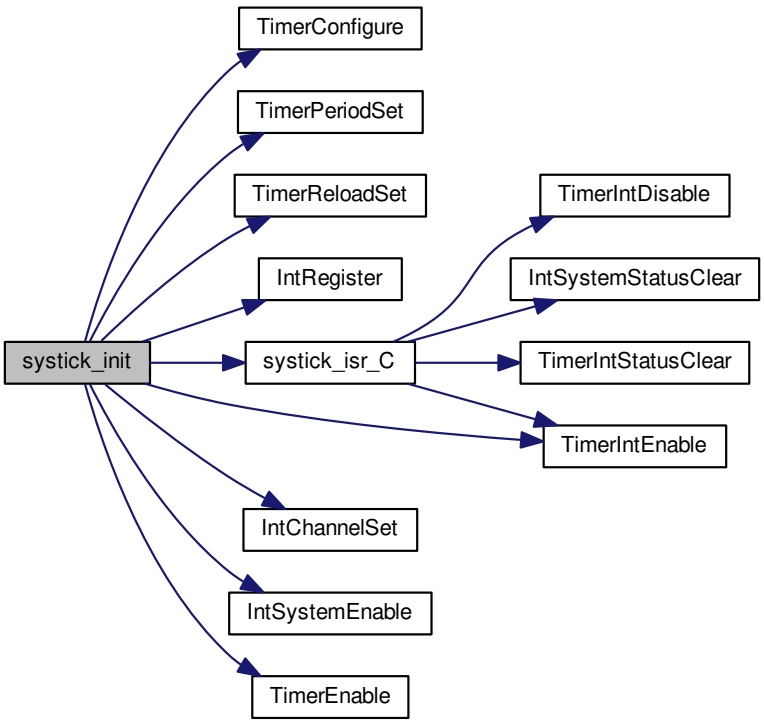
Initialize the systick module, i.e. the hardware timer of the SoC.

This function will register the corresponding ISR, enable the timer interrupt and configure interrupt channel 2 (normal interrupt) for the hardware timer.

**Returns**

none

Here is the call graph for this function:



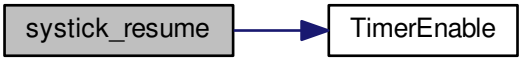

#### 4.66.2.3 void systick_resume ( void )

Enable the timer and therefore the systick.

**Returns**

none

Here is the call graph for this function:



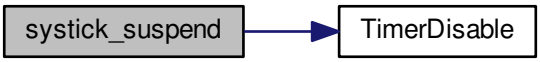

#### 4.66.2.4 void systick_suspend ( void )

Disable the timer and therefore the systick.

**Returns**

none

Here is the call graph for this function:




#### 4.66.2.5 void systick_wait_ms ( U32 *ms* )

Wait for the specified amount of time.

Waiting with this function is an active waiting which will block until the time has elapsed.

**Parameters**

| | |
|---|---|
| *ms* | - The time to wait in milliseconds |

**Returns**

none

#### 4.66.2.6 void systick_wait_ns ( U32 *ns* )

Wait for the specified amount of time.

Waiting with this function is an active waiting which will block until the time has elapsed.

**Parameters**

| | |
|---|---|
| *ns* | - The time to wait in nanoseconds |

**Returns**

none