A20-gpio

Generated by Doxygen 1.8.9.1

Fri Jun 19 2015 16:30:14

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 A20 Namespace Reference

Classes

- class GPIO

  The main class of the module. First of all, you need to call the init() method, to initializate the sunxi driver. After that you can obtain GPIO via get_input, get_output and set_periphery_mode.
- class GPIO_common

  An abstract class for GPIO_input and GPIO_output. Probably, you don't have to use this (internal library use).
- class GPIO_exception

  A generic exception for all module.
- class GPIO_input

  The class that represents a GPIO input. You cannot instantiate this class directly, but you need to call GPIO::get_input to have a pointer of this class. Also, you cannot free the pointer, but you need to use GPIO::free.
- class GPIO_output

  The class that represents a GPIO output. You cannot instantiate this class directly, but you need to call GPIO::get↩ _output to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free.
- class GPIO_periphery

  The class that represents a GPIO periphery. You cannot instantiate this class directly, but you need to call GPIO↩ ::set_periphery_mode to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free. Currently, you cannot do anything with this type of GPIO.

Enumerations

- enum pull_resistor_t { NONE =0, PULL_UP =1, PULL_DOWN =2 }

  The enumeration for pullup/pulldown resistors. The values match sunxi values for pull resistors.

5.1.1 Detailed Description

The namespace for A20

5.1.2 Enumeration Type Documentation

5.1.2.1 enum **A20::pull_resistor_t**

The enumeration for pullup/pulldown resistors. The values match sunxi values for pull resistors.

Enumerator

NONE

PULL_UP

PULL_DOWN

Chapter 6

Class Documentation

6.1 A20::GPIO Class Reference

The main class of the module. First of all, you need to call the init() method, to initializate the sunxi driver. After that you can obtain GPIO via get_input, get_output and set_periphery_mode.

`#include <A20-gpio.hpp>`

Static Public Member Functions

- static void init () throw (GPIO_exception)

  Initialize the submodule sunxi. You have to call this before any other method.

  Exceptions

GPIO_exception	*In case of initialization error.*

- static GPIO_input ∗ get_input (uint16_t port, pull_resistor_t pull_resistor=NONE) throw (GPIO_exception)

  Set the GPIO indicated as input, enable the eventually specified pull resistor, and returns the corresponding object to read data. If you want to change the type of GPIO, you must free the previous object via GPIO::free.
- static GPIO_output ∗ get_output (uint16_t port) throw (GPIO_exception)

  Set the GPIO indicated as output and returns the corresponding object to write data. If you want to change the type of GPIO, you must free the previous object via GPIO::free.
- static void set_periphery_mode (uint16_t port) throw (GPIO_exception)

  Set the GPIO indicated as periphery and returns the corresponding object. If you want to change the type of GPIO, you must free the previous object via GPIO::free.
- static void free (uint16_t port) throw (GPIO_exception)

  Free the corresponding object.

6.1.1 Detailed Description

The main class of the module. First of all, you need to call the init() method, to initializate the sunxi driver. After that you can obtain GPIO via get_input, get_output and set_periphery_mode.

Note

This is a static class, you cannot (and you don't need to) initializate an object of this type.

6.1.2 Member Function Documentation

6.1.2.1 void A20::GPIO::free (uint16_t *port*) throw GPIO_exception) `[static]`

Free the corresponding object.

Parameters

port	the number of GPIO to free according to kernel enumeration (see datasheet)

Note

If the object you ask to free is an output GPIO, this method transform the output in input for precaution. If you don't want this behaviour, define a constant named GPIO_NO_SAFE_RESET_TO_INPUT

Exceptions

GPIO_exception	In case of generic error or if you are trying to free a non setted GPIO.

6.1.2.2 GPIO_input ∗ A20::GPIO::get_input (uint16_t *port*, pull_resistor_t *pull_resistor* = NONE) throw GPIO_exception) [static]

Set the GPIO indicated as input, enable the eventually specified pull resistor, and returns the corresponding object to read data. If you want to change the type of GPIO, you must free the previous object via GPIO::free.

Parameters

port	the number of GPIO to set as input according to kernel enumeration (see datasheet)
pull_resistor	specify the type of pull resistor to enable in this GPIO. Refer to pull_resistor_t for available values. By default no pull resistor is enabled.

Exceptions

GPIO_exception	In case of configuration error or if you are trying to set as input a non freed GPIO (for example because you create a GPIO_output on the same GPIO).

Note

Before you can read from this gpio, you need to wait some microseconds. We already put in this method a sleep of 100us. If you don't want this behaviour, define a costant named GPIO_NO_WAIT.

6.1.2.3 GPIO_output ∗ A20::GPIO::get_output (uint16_t *port*) throw GPIO_exception) [static]

Set the GPIO indicated as output and returns the corresponding object to write data. If you want to change the type of GPIO, you must free the previous object via GPIO::free.

Parameters

port	the number of GPIO to set as output according to kernel enumeration (see datasheet)

Exceptions

GPIO_exception	In case of configuration error or if you are trying to set as output a non freed GPIO (for example because you create a GPIO_input on the same GPIO).

Note

Before you can write to this gpio, you need to wait some microseconds. We already put in this method a sleep of 100us. If you don't want this behaviour, define a costant named GPIO_NO_WAIT.

6.1.2.4 void A20::GPIO::init ( ) throw GPIO_exception) [static]

Initialize the submodule sunxi. You have to call this before any other method.

Exceptions

GPIO_exception	In case of initialization error.

6.1.2.5 void A20::GPIO::set_periphery_mode (uint16_t *port*) throw GPIO_exception) [static]

Set the GPIO indicated as periphery and returns the corresponding object. If you want to change the type of GPIO, you must free the previous object via GPIO::free.

Parameters

port	the number of GPIO to set as periphery according to kernel enumeration (see datasheet)

Exceptions

GPIO_exception	In case of configuration error or if you are trying to set as periphery a non freed GPIO (for example because you create a GPIO_input on the same GPIO).

Note

Before you can use this gpio, you need to wait some microseconds. We already put in this method a sleep of 100us. If you don't want this behaviour, define a costant named GPIO_NO_WAIT.

The documentation for this class was generated from the following files:

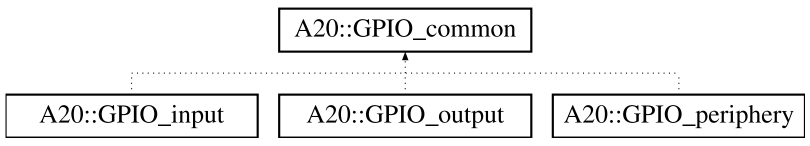
- A20-gpio.hpp
- A20-gpio.cpp

6.2 A20::GPIO_common Class Reference

An abstract class for GPIO_input and GPIO_output. Probably, you don't have to use this (internal library use).

`#include <A20-gpio.hpp>`

Inheritance diagram for A20::GPIO_common:



Public Member Functions

- virtual uint8_t get_type () const =0

  It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).
- virtual uint16_t get_port () const =0

  It returns the current port number as requested in costructor.

6.2.1 Detailed Description

An abstract class for GPIO_input and GPIO_output. Probably, you don't have to use this (internal library use).

6.2.2 Member Function Documentation

6.2.2.1 virtual uint16_t A20::GPIO_common::get_port ( ) const [pure virtual]

It returns the current port number as requested in costructor.

Returns

the port number according to linux enumeration (see datasheet)

Implemented in A20::GPIO_periphery, A20::GPIO_output, and A20::GPIO_input.

6.2.2.2 virtual uint8_t A20::GPIO_common::get_type ( ) const [pure virtual]

It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).

Returns

0 if input, 1 if output, 2 if periphery

Implemented in A20::GPIO_periphery, A20::GPIO_output, and A20::GPIO_input.

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

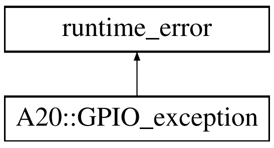
- A20-gpio.hpp

6.3 A20::GPIO_exception Class Reference

A generic exception for all module.

`#include <A20-gpio.hpp>`

Inheritance diagram for A20::GPIO_exception:



Public Member Functions

- GPIO_exception (std::string s)

6.3.1 Detailed Description

A generic exception for all module.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 A20::GPIO_exception::GPIO_exception (std::string *s*) [inline]

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

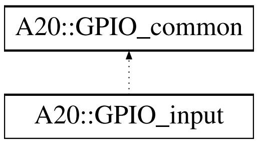
- A20-gpio.hpp

6.4 A20::GPIO_input Class Reference

The class that represents a GPIO input. You cannot instantiate this class directly, but you need to call GPIO::get↩
_input to have a pointer of this class. Also, you cannot free the pointer, but you need to use GPIO::free.

`#include <A20-gpio.hpp>`

Inheritance diagram for A20::GPIO_input:



Public Member Functions

- uint8_t get_type () const

  It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).
- uint16_t get_port () const

  It returns the current port number as requested in costructor.
- bool get () const

  It returns the current value of the GPIO.

Friends

- class GPIO

6.4.1 Detailed Description

The class that represents a GPIO input. You cannot instantiate this class directly, but you need to call GPIO::get↩
_input to have a pointer of this class. Also, you cannot free the pointer, but you need to use GPIO::free.

6.4.2 Member Function Documentation

6.4.2.1 bool A20::GPIO_input::get ( ) const

It returns the current value of the GPIO.

Returns

true if it reads a logical '1' on the GPIO, false otherwise.

6.4.2.2 uint16_t A20::GPIO_input::get_port ( ) const `[inline], [virtual]`

It returns the current port number as requested in costructor.

Returns

the port number according to linux enumeration (see datasheet)

Implements A20::GPIO_common.

6.4.2.3 uint8_t A20::GPIO_input::get_type ( ) const [inline], [virtual]

It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).

Returns

0 if input, 1 if output, 2 if periphery

Implements A20::GPIO_common.

6.4.3 Friends And Related Function Documentation

6.4.3.1 friend class GPIO [friend]

The documentation for this class was generated from the following files:

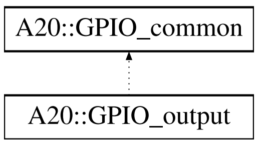
- A20-gpio.hpp
- A20-gpio.cpp

6.5 A20::GPIO_output Class Reference

The class that represents a GPIO output. You cannot instantiate this class directly, but you need to call GPI↩ O::get_output to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free.

```
#include <A20-gpio.hpp>
```

Inheritance diagram for A20::GPIO_output:



Public Member Functions

- uint8_t get_type () const

  It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).
- uint16_t get_port () const

  It returns the current port number as requested in costructor.
- void set (bool) const

  Set the logical output of a GPIO.

Friends

- class GPIO

6.5.1 Detailed Description

The class that represents a GPIO output. You cannot instantiate this class directly, but you need to call GPI↩ O::get_output to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free.

6.5.2 Member Function Documentation

6.5.2.1 uint16_t A20::GPIO_output::get_port ( ) const [inline], [virtual]

It returns the current port number as requested in costructor.

Returns

the port number according to linux enumeration (see datasheet)

Implements A20::GPIO_common.

6.5.2.2 uint8_t A20::GPIO_output::get_type ( ) const [inline], [virtual]

It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).

Returns

0 if input, 1 if output, 2 if periphery

Implements A20::GPIO_common.

6.5.2.3 void A20::GPIO_output::set (bool *value*) const

Set the logical output of a GPIO.

Parameters

value	if true it puts logical '1' on the GPIO, '0' otherwise.

6.5.3 Friends And Related Function Documentation

6.5.3.1 friend class GPIO [friend]

The documentation for this class was generated from the following files:

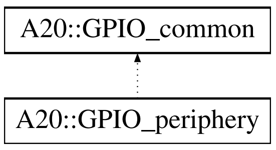
- A20-gpio.hpp
- A20-gpio.cpp

6.6 A20::GPIO_periphery Class Reference

The class that represents a GPIO periphery. You cannot instantiate this class directly, but you need to call GPIO↵::set_periphery_mode to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free. Currently, you cannot do anything with this type of GPIO.

```
#include <A20-gpio.hpp>
```

Inheritance diagram for A20::GPIO_periphery:



Public Member Functions

- uint8_t get_type () const

  It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).
- uint16_t get_port () const

  It returns the current port number as requested in costructor.

Friends

- class GPIO

6.6.1 Detailed Description

The class that represents a GPIO periphery. You cannot instantiate this class directly, but you need to call GPIO↩
::set_periphery_mode to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free. Currently, you cannot do anything with this type of GPIO.

6.6.2 Member Function Documentation

6.6.2.1 uint16_t A20::GPIO_periphery::get_port ( ) const [inline], [virtual]

It returns the current port number as requested in costructor.

Returns

the port number according to linux enumeration (see datasheet)

Implements A20::GPIO_common.

6.6.2.2 uint8_t A20::GPIO_periphery::get_type ( ) const [inline], [virtual]

It returns the type of GPIO (input, output or periphery). Probably, you don't have to use this (internal library use).

Returns

0 if input, 1 if output, 2 if periphery

Implements A20::GPIO_common.

6.6.3 Friends And Related Function Documentation

6.6.3.1 friend class GPIO [friend]

The documentation for this class was generated from the following files:

- A20-gpio.hpp
- A20-gpio.cpp

Chapter 7

File Documentation

7.1 A20-gpio.cpp File Reference

```
#include "A20-gpio.hpp"
#include "gpio_lib.h"
#include <unistd.h>
```

Namespaces

- A20

7.2 A20-gpio.hpp File Reference

```
#include <vector>
#include <stdexcept>
#include <string>
```

Classes

- class A20::GPIO_exception

  A generic exception for all module.
- class A20::GPIO_common

  An abstract class for GPIO_input and GPIO_output. Probably, you don't have to use this (internal library use).
- class A20::GPIO_input

  The class that represents a GPIO input. You cannot instantiate this class directly, but you need to call GPIO::get_input to have a pointer of this class. Also, you cannot free the pointer, but you need to use GPIO::free.
- class A20::GPIO_output

  The class that represents a GPIO output. You cannot instantiate this class directly, but you need to call GPIO::get↩ _output to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free.
- class A20::GPIO_periphery

  The class that represents a GPIO periphery. You cannot instantiate this class directly, but you need to call GPIO↩ ::set_periphery_mode to have a pointer of an object of this type. Also, you cannot free the pointer, but you need to use GPIO::free. Currently, you cannot do anything with this type of GPIO.
- class A20::GPIO

  The main class of the module. First of all, you need to call the init() method, to initializate the sunxi driver. After that you can obtain GPIO via get_input, get_output and set_periphery_mode.

Namespaces

- A20

Typedefs

- typedef unsigned char uint8_t
- typedef unsigned short uint16_t

Enumerations

- enum A20::pull_resistor_t { A20::NONE =0, A20::PULL_UP =1, A20::PULL_DOWN =2 }

  The enumeration for pullup/pulldown resistors. The values match sunxi values for pull resistors.

7.2.1 Typedef Documentation

7.2.1.1 typedef unsigned short **uint16_t**

7.2.1.2 typedef unsigned char **uint8_t**

7.3 examples/ex.cpp File Reference

```
#include "../A20-gpio.hpp"
#include <assert.h>
#include <iostream>
```

Functions

- int main ()

7.3.1 Function Documentation

7.3.1.1 int main ( )