# C-API to access MCU periphals

DIN Rail Controller DRC02, BigFish

DH electronics GmbH - Am Anger 8 - 83346 Bergen - Germany

YOUR DIGITAL HEROES.

# Contents

**1 C-API - a library to abstract the SPI communication to the MCU for peripheral expansion. 1**

1.0.1 Usage . . . 2
1.0.2 Abbreviations . . . 2
1.0.3 C-API Reference . . . 2
1.0.4 C++ API's and Dependencies . . . 3
1.0.5 Compiling . . . 3
1.0.6 License . . . 4

**2 Data Structure Index 5**

2.0.1 Data Structures . . . 5

**3 File Index 6**

3.0.1 File List . . . 6

**4 Data Structure Documentation 7**

4.0.1 Bitmap Struct Reference . . . 7
4.0.2 FT_Bitmap_ Struct Reference . . . 9

**5 File Documentation 12**

5.0.1 bitmap.h File Reference . . . 12
5.0.2 buttons.h File Reference . . . 13
5.0.3 c-api.h File Reference . . . 15
5.0.4 display.h File Reference . . . 16
5.0.5 helpers.h File Reference . . . 23
5.0.6 init.h File Reference . . . 24
5.0.7 leds.h File Reference . . . 26
5.0.8 resultcodes.h File Reference . . . 28
5.0.9 stats.h File Reference . . . 29

**Index 34**

Chapter 1

# C-API - a library to abstract the SPI communication to the MCU for peripheral expansion.

The DRC02 and BigFish is driven by a DHCOM module and an I/O extension by a MCU. The DRC02 has a Cypress PSoC MCU and the BigFish periphals are handled by a Microchip PIC. The MCU is controlled by the DHCOM module through a SPI interface (DHCOM is SPI Master, MCU is SPI slave) in both cases.

Connected to MCU (PSoC/PIC):

- 5 capacitive buttons
- OLED Display (via SPI)
- 4 LEDs

The PSoC/PIC takes care of the control of the display, the PCAP keys query and the LED control. To simplify the communication to these devices for software developers a software library have been developed to offer a simple API that abstracts the SPI protocol between DHCOM and the Cypress MCU. The customer application is compiled against C-API and uses direct calls to API functions.

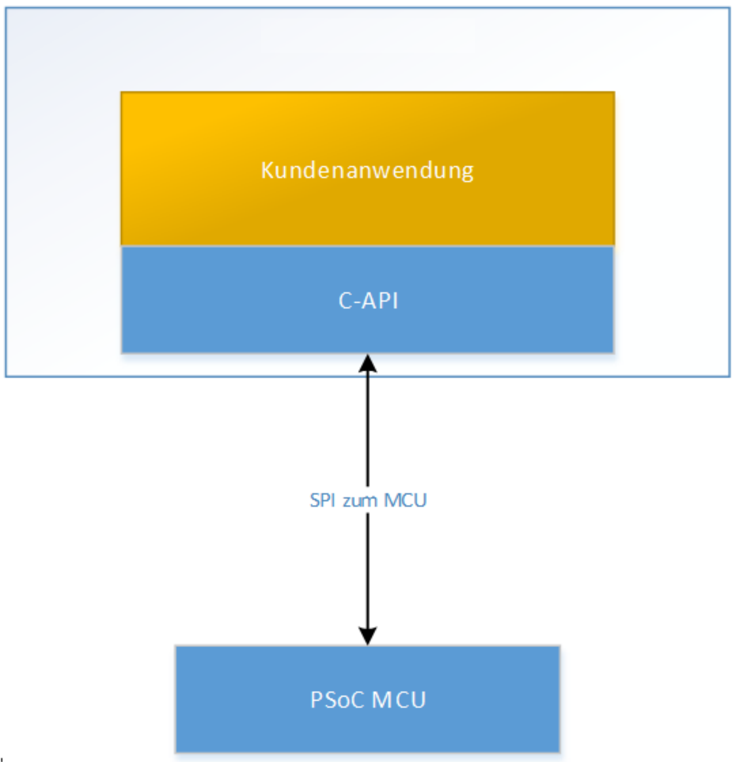


**Figure 1.1 Software structure concerning customers application**

This documentation describes only the public API of this library. No information regarding internal implementation is given.

### 1.0.1 Usage

To get full access to the public API of this library only the header file c-api.h has to be included:

#include <c-api/c-api.h>

### 1.0.2 Abbreviations

**Table 1.1 abbrevations used in this document**

| abb. | explanation |
|---|---|
| LAN | Local Area Network |
| API | Application Programming Interface |
| PCAP | Projected Capacitance |
| LED | Light Emitting Diode |
| SW | Software |
| HW | Hardware |
| HAL | Hardware Abstraction Layer |

### 1.0.3 C-API Reference

C-API methods have the following common properties:

- methods return the results (error codes);
- methods do not show C++ exceptions;
- methods with prefix get- or set- are fast, and return immediately (non blocking), because they only return local data.
- Methods with prefix write- or read- cannot return immediately (blocking) until the SPI comcommunication is terminated.
- All methods are reentrant, which means that they can also be called by multiple threads.

### 1.0.4 C++ API's and Dependencies

#### 1.0.4.1 DHCOM_HAL

A HAL abstraction library for DHCOM modules DHCOM_HAL is used and included. If you have any questions, please contact us.

#### 1.0.4.2 Text API

To draw the TTF texts we use an existing API 'FreeType' from `https://www.freetype.org/` in the form of the dynamic library (due to GPL license). The API has all needed functions, like:

- Selection of the font
- Selection color of the font
- Estimate the size of the border rectangle for certain text
- Drawing of the glyph in the FT_Bitmaps

To draw prepared glyphs on the bitmap the Bitmap::blit method is used. Then the bitmap can be sent to the screen using C-API.

### 1.0.5 Compiling

It is intended to compile the C-API with CMAKE build system.

- clone or checkout source into local directory 'projectdir':

Create a out of source tree build directory:

- cd projectdir/..
- mkdir build
- cd build/

To use the yocto toolchain you need to source the environment setup script before running cmake:

- . /opt/fslc-framebuffer/2.6.2/environment-setup-armv7at2hf-neon-fslc-linux-gnueabi

To setup the project for a Eclipse IDE do:

- cmake CMakeLists.txt -G "Eclipse CDT4 - Unix Makefiles" ../psoc-host-application/
- start Eclipse and use import dialog to open the directory 'build'
- create a build target which just calls make

### 1.0.6 License

# Chapter 2

# Data Structure Index

### 2.0.1 Data Structures

Here are the data structures with brief descriptions:

Bitmap
Structure for handling image information . . . . . . . . . . . . . . . . . . . . . . . . . . 7
FT_Bitmap_
Bitmap based on FreeType API . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Chapter 3

# File Index

### 3.0.1 File List

Here is a list of all files with brief descriptions:

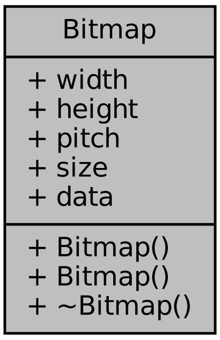
bitmap.h
Header for Bitmap definitions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
buttons.h
Header for accessing the PCAP buttons . . . . . . . . . . . . . . . . . . . . . . . . . . 13
c-api.h
Header for the full C-API include . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
display.h
Header for API to control the OLED Display . . . . . . . . . . . . . . . . . . . . . . . 16
helpers.h
Helpers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
init.h
API initialization and termination . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
leds.h
Header for LED control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
resultcodes.h
Header for result code handling . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
stats.h
Handle SPI communication statistics . . . . . . . . . . . . . . . . . . . . . . . . . . 29

Chapter 4

# Data Structure Documentation

### 4.0.1 Bitmap Struct Reference

Structure for handling image information.
`#include <bitmap.h>`
Collaboration diagram for Bitmap:



**Public Member Functions**

- Bitmap (int w, int h, const unsigned char ∗bytes=NULL)
- Bitmap (const Bitmap &other)
- ∼Bitmap ()

**Data Fields**

- const int width

  *visible width [pixel]*
- const int height

  *visible height [lines/pixel]*
- const int pitch

  *The pitchs value is the number of bytes taken by one bitmap row, including padding. [pixel].*
- const int size

  *size of data buffer [bytes]*
- unsigned char ∗ data

  *data buffer*

#### 4.0.1.1 Detailed Description

Structure for handling image information.

A structure used to describe a bitmap or pixmap

Attention

The bitmap must contain one bit per pixel and the bits must be arranged in lines. Line breaks are made on byte boundaries (pitch bits per line), although the width of the visible bitmap does not always fall exactly to the byte boundary (width bits). The remaining (pitch) bits at the end of the line are ignored.

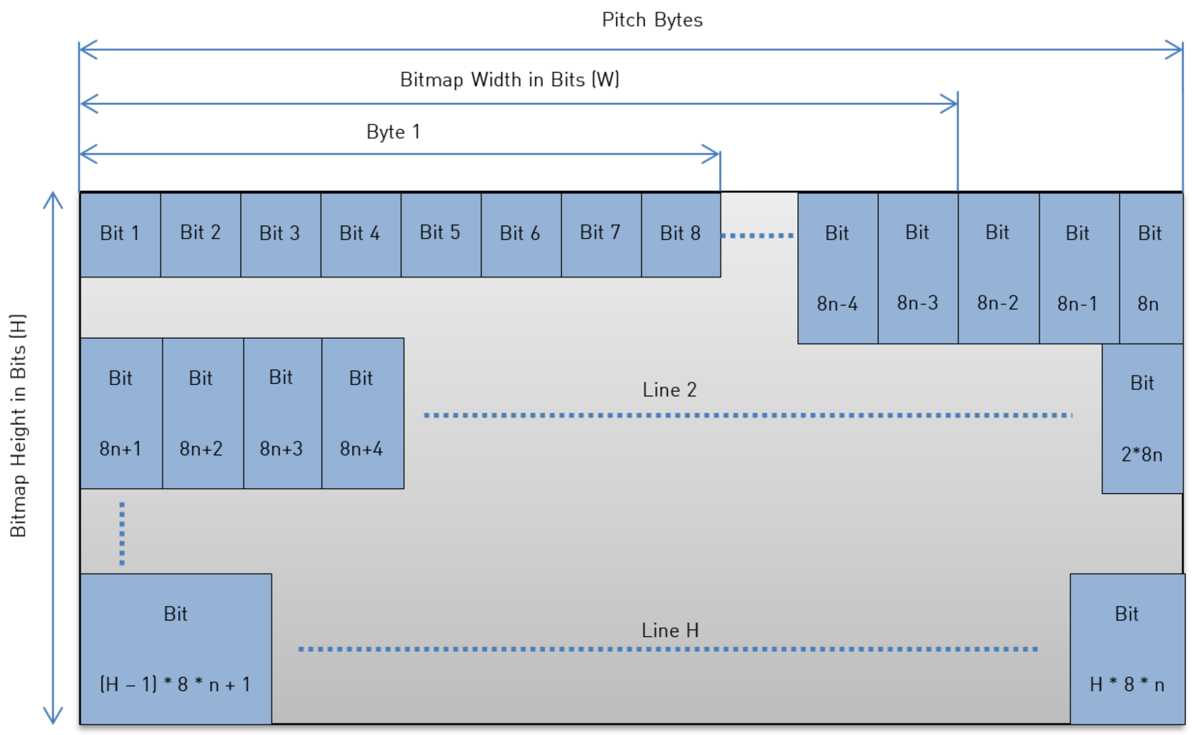


**Figure 4.1 Bitmap data arrangement**

#### 4.0.1.2 Constructor & Destructor Documentation

**Bitmap()** [1/2]
```
Bitmap::Bitmap (
    int w,
    int h,
    const unsigned char * bytes = NULL )
```

**Bitmap()** [2/2]
```
Bitmap::Bitmap (
    const Bitmap & other )
```

**~Bitmap()** `Bitmap::~Bitmap ( )`

#### 4.0.1.3 Field Documentation

**width** `const int Bitmap::width`

visible width [pixel]

**height** `const int Bitmap::height`

visible height [lines/pixel]

**pitch** `const int Bitmap::pitch`

The pitchs value is the number of bytes taken by one bitmap row, including padding. [pixel].

**size** `const int Bitmap::size`

size of data buffer [bytes]

**data** `unsigned char* Bitmap::data`

data buffer

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

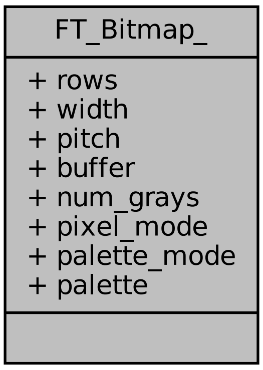
- bitmap.h

### 4.0.2 FT_Bitmap_ Struct Reference

Bitmap based on FreeType API.

`#include <bitmap.h>`

Collaboration diagram for FT_Bitmap_:



### Data Fields

- unsigned int rows

  *The number of bitmap rows. [lines/pixel].*
- unsigned int width

  *The number of pixels in bitmap row. [pixel].*
- int pitch

  *The pitchs value is the number of bytes taken by one bitmap row, including padding. [pixel].*
- unsigned char ∗ buffer

  *A typeless pointer to the bitmap buffer. This value should be aligned on 8-bit boundaries.*
- unsigned short num_grays

  *This field is only used with FT_PIXEL_MODE_GRAY; it gives the number of gray levels used in the bitmap.*
- unsigned char pixel_mode

  *The pixel mode, i.e., how pixel bits are stored. See FT_Pixel_Mode for possible values.*
- unsigned char palette_mode

  *This field is intended for paletted pixel modes; it indicates how the palette is stored. Not used currently.*
- void ∗ palette

  *A typeless pointer to the bitmap palette; this field is intended for paletted pixel modes. Not used currently.*

#### 4.0.2.1 Detailed Description

Bitmap based on FreeType API.

A structure used to describe a bitmap or pixmap. See

- `https://www.freetype.org/freetype2/docs/reference/ft2-bitmap_handling.html` and
- `https://www.freetype.org/freetype2/docs/reference/ft2-basic_types.html#ft_bitmap`

for further information.

#### 4.0.2.2 Field Documentation

**rows** `unsigned int FT_Bitmap_::rows`

The number of bitmap rows. [lines/pixel].

**width** `unsigned int FT_Bitmap_::width`

The number of pixels in bitmap row. [pixel].

**pitch** `int FT_Bitmap_::pitch`

The pitchs value is the number of bytes taken by one bitmap row, including padding. [pixel].

**buffer** `unsigned char* FT_Bitmap_::buffer`

A typeless pointer to the bitmap buffer. This value should be aligned on 8-bit boundaries.

**num_grays** `unsigned short FT_Bitmap_::num_grays`

This field is only used with FT_PIXEL_MODE_GRAY; it gives the number of gray levels used in the bitmap.

**pixel_mode** `unsigned char FT_Bitmap_::pixel_mode`

The pixel mode, i.e., how pixel bits are stored. See FT_Pixel_Mode for possible values.

**palette_mode** `unsigned char FT_Bitmap_::palette_mode`

This field is intended for paletted pixel modes; it indicates how the palette is stored. Not used currently.

**palette** `void* FT_Bitmap_::palette`

A typeless pointer to the bitmap palette; this field is intended for paletted pixel modes. Not used currently.

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

- bitmap.h

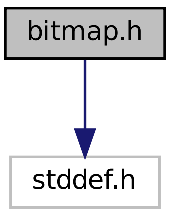
Chapter 5

# File Documentation

### 5.0.1 bitmap.h File Reference

Header for Bitmap definitions.

`#include <stddef.h>`
Include dependency graph for bitmap.h:



**Data Structures**

- struct Bitmap

    *Structure for handling image information.*

- struct FT_Bitmap_

    *Bitmap based on FreeType API.*

#### 5.0.1.1 Detailed Description

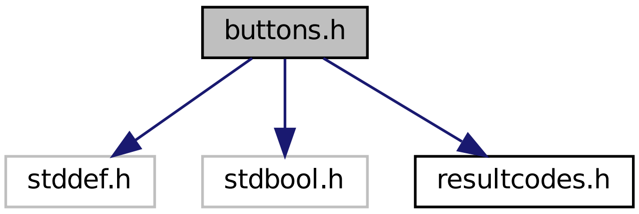
Header for Bitmap definitions.

### 5.0.2 buttons.h File Reference

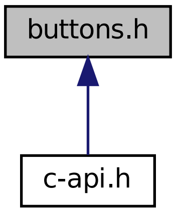
Header for accessing the PCAP buttons.

```
#include <stddef.h>
#include <stdbool.h>
#include "resultcodes.h"
```

Include dependency graph for buttons.h:



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



#### Typedefs

- typedef void buttonsCallback(enum BUTTON button, bool pressed)

#### Enumerations

- enum BUTTON {
  BUTTON_DN_LEFT = 0 , BUTTON_DN_RIGHT = 1 , BUTTON_MID = 2 , BUTTON_UP_LEFT = 3 ,
  BUTTON_UP_RIGHT = 4 }

#### Functions

- enum RESULT setButtonsCallback (buttonsCallback *callback)

  *set callback for PCAP buttons*
- enum RESULT handleButtons ()

  *trigger PCAP buttons measurement*
- bool getButtonState (enum BUTTON button, enum RESULT *res=NULL)

  *read state of a specific PCAP button*

#### 5.0.2.1 Detailed Description

Header for accessing the PCAP buttons.

#### 5.0.2.2 Typedef Documentation

**buttonsCallback** `typedef void buttonsCallback(enum BUTTON button, bool pressed)`

#### 5.0.2.3 Enumeration Type Documentation

**BUTTON** `enum BUTTON`

cap. touch buttons of DRC02/Bigfish front panel

Enumerator

| | |
|---|---|
| BUTTON_DN_LEFT | DN_LEFT (left) button. |
| BUTTON_DN_RIGHT | DN_RIGHT (right) button. |
| BUTTON_MID | MID (OK) button. |
| BUTTON_UP_LEFT | UP_LEFT (up) button. |
| BUTTON_UP_RIGHT | UP_RIGHT (down) button. |

#### 5.0.2.4 Function Documentation

**setButtonsCallback()**
```
enum RESULT setButtonsCallback (
        buttonsCallback * callback )
```

set callback for PCAP buttons

Sets a new callback for PCAP buttons. Can also be called with NULL. If not NULL the passed function is called for each PCAP button event (pressed and released).

Parameters

| | |
|---|---|
| *callback* | function pointer to callback function |

Returns

result code of requested operation

**handleButtons()** `enum RESULT handleButtons ( )`

trigger PCAP buttons measurement

Reads the PCAP buttons states, therefore it must be called regularly. When PCAP buttons states are changing, the callback for each state change of each PCAP button is called.

Returns

result code of requested operation

**getButtonState()**

```
bool getButtonState (
            enum BUTTON button,
            enum RESULT * res = NULL )
```

read state of a specific PCAP button

For most applications it is recommended to work with handleButtons() and a callback instead of using getButtonState().

Parameters

| | |
|---|---|
| *button* | pass the PCAP button of interest |
| *res* | pointer to pass RESULT of operation |

Returns

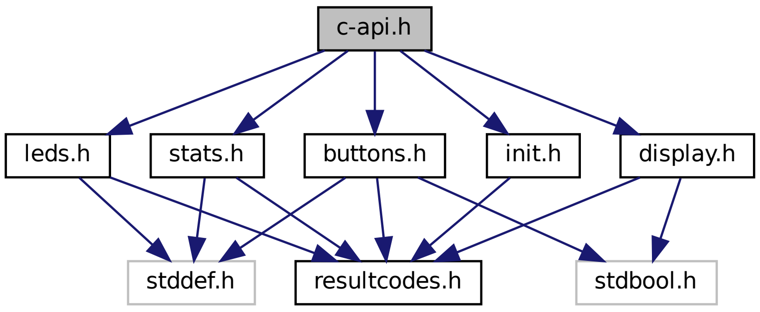
state of button (true==pressed/false==not pressed)

### 5.0.3 c-api.h File Reference

Header for the full C-API include.

```
#include "init.h"
#include "stats.h"
#include "leds.h"
#include "display.h"
#include "buttons.h"
```

Include dependency graph for c-api.h:



#### 5.0.3.1 Detailed Description

Header for the full C-API include.

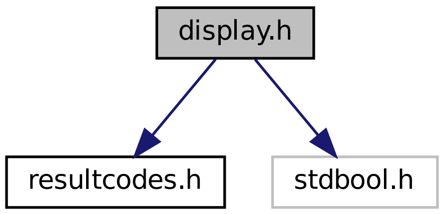
Applications which link against the C-API library only have to include this header file.

### 5.0.4 display.h File Reference

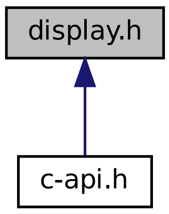
Header for API to control the OLED Display.

```
#include "resultcodes.h"
#include <stdbool.h>
```

Include dependency graph for display.h:



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



**Macros**

- #define DISPLAY_DEFAULT_WIDTH 128

  *display resolution in x-direction*
- #define DISPLAY_DEFAULT_HEIGHT 64

  *display resolution in y-direction*

**Functions**

- enum RESULT displayEnable (int on)
  *enable/disable the display*
- enum RESULT displaySetContrast (int value)
  *set display contrast*
- enum RESULT displaySetDimTimeout (short value)
  *set display dimming timeout*
- enum RESULT displaySetOffTimeout (short value)
  *set display switch off timeout*
- enum RESULT displayFill (bool white=false)
  *fill display with single color*
- enum RESULT displayFillRect (int x, int y, int w, int h, bool white=true)
  *draw a filled rectangle*
- enum RESULT displayDrawRect (int x, int y, int w, int h, bool white=true)
  *draw a rectangle*
- enum RESULT displayInvertRect (int x, int y, int w, int h)
  *invert pixels in rectangle*
- enum RESULT displayBitmap (int x, int y, const Bitmap ∗bmp)
  *write bitmap*
- enum RESULT displayBitmap2 (int x, int y, const struct FT_Bitmap_ ∗bmp)
  *write bitmap*
- enum RESULT displayScreen (const unsigned char ∗screen_buffer)
  *write buffer content to (full) screen*
- enum RESULT displayFlush ()
  *write display frame to OLED display via MCU*
- enum RESULT displaySwap ()
  *re-send the last compressed buffer*
- enum RESULT displayWriteSplash ()
  *write splash*
- enum RESULT displayTakeScreenshot ()
  *take BMP screenshot*

### 5.0.4.1 Detailed Description

Header for API to control the OLED Display.

The OLED display is connected to the MCU with SPI.

To avoid wearing of the OLED display, it is dimmed after a first timeout and the output is deactivated after another timeout. The OLED display can only be reactivated by touching a PCAP button on the front of the device. Touching a PCAP button also resets the timeouts. See displaySetDimTimeout() and displaySetOffTimeout().

### 5.0.4.2 Macro Definition Documentation

**DISPLAY_DEFAULT_WIDTH** `#define DISPLAY_DEFAULT_WIDTH 128`

display resolution in x-direction

Resolution of the OLED SSD1306 in x-direction. Used in displayScreen().

**DISPLAY_DEFAULT_HEIGHT** `#define DISPLAY_DEFAULT_HEIGHT 64`

display resolution in y-direction

Resolution of the OLED SSD1306 in y-direction. Used in displayScreen().

### 5.0.4.3 Function Documentation

**displayEnable()**

```
enum RESULT displayEnable (
            int on )
```

enable/disable the display

Switch the display enable output of the MCU.

Attention

This is only available in BigFish hardware with PIC. It does set the OLED::Reset line (low active). DRC02 Hardware with PSoC does return OK to be legacy compatible but without any action.

Parameters

| | |
|---|---|
| *on* | true == enabled / false == disabled |

Returns

result code of requested operation

**displaySetContrast()**

```
enum RESULT displaySetContrast (
            int value )
```

set display contrast

Attention

This feature is blocked by latest MCU firmwares. The contrast values are handled by the MCU firmware and can not be adjusted. See displaySetDimTimeout() and displaySetOffTimeout() for further details.

Parameters

| | |
|---|---|
| *value* | new display contrast value |

Returns

result code of requested operation

#### displaySetDimTimeout()

```
enum RESULT displaySetDimTimeout (
        short value )
```

set display dimming timeout

After this time the brightness of the OLED display is reduced. The timeout is reseted or/and the display reactivated/brightness increased by touching the PCAP buttons. This serves to protect the OLED display. The default value is 300 seconds. The maximum value is 300 seconds.

Parameters

| | |
|---|---|
| *value* | new display dimming timeout [sec] |

Returns

result code of requested operation

#### displaySetOffTimeout()

```
enum RESULT displaySetOffTimeout (
        short value )
```

set display switch off timeout

This timeout starts after the dimming timeout expired and the display brightness was decreased. The switch off timeout does disable the display. The timeout is reseted or/and the display reactivated/brightness increased by touching the PCAP buttons. This serves to protect the OLED display. The default value is 600 seconds. The maximum value is 600 seconds.

Parameters

| | |
|---|---|
| *value* | new display switch off timeout [sec] |

Returns

result code of requested operation

#### displayFill()

```
enum RESULT displayFill (
        bool white = false )
```

fill display with single color

Parameters

| | |
|---|---|
| *white* | if true than fill display with white (foreground color) |

Returns

result code of requested operation

**displayFillRect()**

```
enum RESULT displayFillRect (
            int x,
            int y,
            int w,
            int h,
            bool white = true )
```

draw a filled rectangle

Parameters

| | |
|---|---|
| *x* | coordinate [pixel] |
| *y* | coordinate [pixel] |
| *w* | width [pixel] |
| *h* | height [pixel] |
| *white* | foreground (true) or background color (false) |

Returns

result code of requested operation

**displayDrawRect()**

```
enum RESULT displayDrawRect (
            int x,
            int y,
            int w,
            int h,
            bool white = true )
```

draw a rectangle

Parameters

| | |
|---|---|
| *x* | coordinate [pixel] |
| *y* | coordinate [pixel] |
| *w* | width [pixel] |
| *h* | height [pixel] |
| *white* | foreground (true) or background color (false) |

Returns

result code of requested operation

**displayInvertRect()**

```
enum RESULT displayInvertRect (
            int x,
            int y,
            int w,
            int h )
```

invert pixels in rectangle

Parameters

| | |
|---|---|
| *x* | coordinate [pixel] |
| *y* | coordinate [pixel] |
| *w* | width [pixel] |
| *h* | height [pixel] |

Returns

result code of requested operation

**displayBitmap()**
```
enum RESULT displayBitmap (
        int x,
        int y,
        const Bitmap * bmp )
```

write bitmap

Parameters

| | |
|---|---|
| *x* | coordinate [pixel] |
| *y* | coordinate [pixel] |
| *bmp* | pointer to pixmap/bitmap |

Returns

result code of requested operation

**displayBitmap2()**
```
enum RESULT displayBitmap2 (
        int x,
        int y,
        const struct FT_Bitmap_ * bmp )
```

write bitmap

Parameters

| | |
|---|---|
| *x* | coordinate [pixel] |
| *y* | coordinate [pixel] |
| *bmp* | pointer to pixmap/bitmap |

Returns

result code of requested operation

**displayScreen()** `enum RESULT displayScreen ( const unsigned char * screen_buffer )`

write buffer content to (full) screen

Parameters

| | |
|---|---|
| *screen_buffer* | pointer to buffer with content to display |

For the resolution of the OLED SSD1306, the transferred buffer must contain data for a full screen. See DISPLAY_DEFAULT_WIDTH and DISPLAY_DEFAULT_HEIGHT. For each pixel there must be at least one bit in the buffer.

Returns

result code of requested operation

**displayFlush()** `enum RESULT displayFlush ( )`

write display frame to OLED display via MCU

Transmit the current frame to the OLED display.

Returns

result code of requested operation

**displaySwap()** `enum RESULT displaySwap ( )`

re-send the last compressed buffer

Returns

result code of requested operation

**displayWriteSplash()** `enum RESULT displayWriteSplash ( )`

write splash

Write the current compressed buffer into MCU flash

Returns

result code of requested operation

**displayTakeScreenshot()** `enum RESULT displayTakeScreenshot ( )`

take BMP screenshot

Take screenshot of OLED and write to BMP file

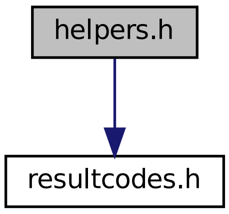
Returns

result code of requested operation

### 5.0.5 helpers.h File Reference

Helpers.

`#include "resultcodes.h"`
Include dependency graph for helpers.h:



**Functions**

- const char ∗ getResultCodeString (RESULT result)
  *Translate enum RESULT in a printable char string.*

#### 5.0.5.1 Detailed Description

Helpers.

#### 5.0.5.2 Function Documentation

**getResultCodeString()** `const char* getResultCodeString ( RESULT result )`

Translate enum RESULT in a printable char string.

Parameters

| | |
|---|---|
| *result* | - pass the returned result of a C-API call |

Returns

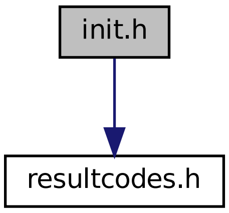
pointer to char string

### 5.0.6 init.h File Reference

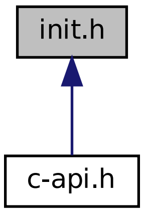
API initialization and termination.

`#include "resultcodes.h"`

Include dependency graph for init.h:



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



**Functions**

- enum RESULT openApi ()

  *open API*
- enum RESULT closeApi ()

  *close API*
- enum RESULT idleApi ()

  *idle API*

#### 5.0.6.1 Detailed Description

API initialization and termination.

Before the Linux operating system boots and the Cypress MCU is accessed by the user application, Green RUN LED will flash with 800ms interval. This ends when user application opens API: Green LED will stop flashing and the control over the Led is transferred to the application.

After the idleAPI call, C-API takes over the flashing and flashes with 300ms interval.

Note: Green RUN LED will stop flashing when first API call is executed. The user application software must then explicitly control the Green RUN LED.

#### 5.0.6.2 Function Documentation

**openApi()** `enum RESULT openApi ( )`

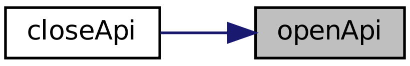
open API

opens API, brings the Cypress MCU to active mode

Returns

    result code of requested operation

Here is the caller graph for this function:



**closeApi()** `enum RESULT closeApi ( )`

close API

Brings Cypress MCU back to wait mode, cleans up resources. The RUN LED will start flashing with 800ms interval, again. The pcap touch button scan is disabled.

Returns

    result code of requested operation

**idleApi()** `enum RESULT idleApi ( )`

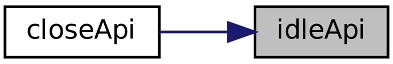
idle API

RUN LED will start flashing with 300mS interval. The pcap touch button scan is disabled during idle mode. (Just don't use idleApi() it if pcap touch buttons should always be active.)

Returns

result code of requested operation

Here is the caller graph for this function:

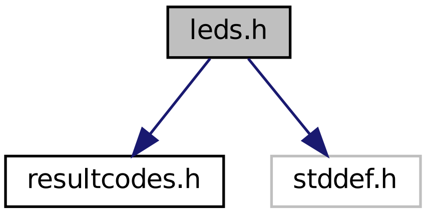


### 5.0.7 leds.h File Reference

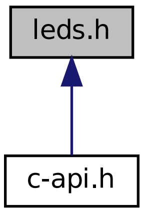
Header for LED control.

```
#include "resultcodes.h"
#include <stddef.h>
```

Include dependency graph for leds.h:



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



#### Enumerations

- enum LED { LED_RUN , LED_LAN , LED_BUS , LED_ERR }

#### Functions

- enum RESULT writeLed (enum LED led, int on)
  *control a LED*
- int getLedState (enum LED led, enum RESULT ∗result=NULL)
  *read state of LED*

### 5.0.7.1 Detailed Description

Header for LED control.

### 5.0.7.2 Enumeration Type Documentation

**LED** `enum LED`

LEDs in DRC02 front panel

Enumerator

| | |
|---|---|
| LED_RUN | run LED (green) |
| LED_LAN | bus LED (green) |
| LED_BUS | lan LED (green) |
| LED_ERR | err LED (red) |

### 5.0.7.3 Function Documentation

**writeLed()**
```
enum RESULT writeLed (
    enum LED led,
    int on )
```

control a LED

Switch on/switch off a LED

Parameters

| | |
|---|---|
| *led* | choose LED to control |
| *on* | pass requested state (on=!0 / off==0) of LED |

Returns

result code of requested operation

**getLedState()**

```
int getLedState (
        enum LED led,
        enum RESULT * result = NULL )
```

read state of LED

Check if a LED is on/off.

Parameters

| | |
|---|---|
| *led* | choose LED to control |
| *res* | pointer to pass RESULT of operation |

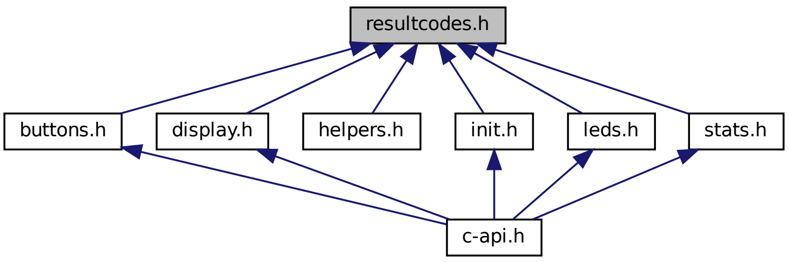
Returns

current state of LED (on=!0 / off==0)

### 5.0.8 resultcodes.h File Reference

Header for result code handling.

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



**Enumerations**

- enum RESULT {
  RESULT_OK , RESULT_API_NOT_OPEN , RESULT_OBJECT_INVALID , RESULT_ARGUMENT_INVALID
  ,
  RESULT_FUNCTION_INVALID , RESULT_SERVICE_INVALID , RESULT_COMM_TIMEOUT ,
  RESULT_SEE_ERRNO }

  *Enumeration for common return codes from **C-API** methods.*

#### 5.0.8.1 Detailed Description

Header for result code handling.

#### 5.0.8.2 Enumeration Type Documentation

**RESULT** `enum RESULT`

Enumeration for common return codes from **C-API** methods.

Most of the class methods in the library return the RESULT code to reflect the success of the requested operation.

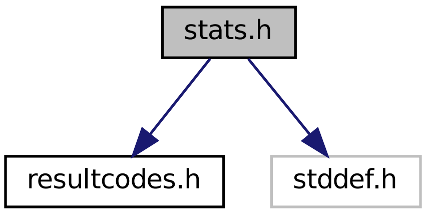
Enumerator

| | |
|---|---|
| RESULT_OK | successfully executed |
| RESULT_API_NOT_OPEN | C-API not open, use openApi() |
| RESULT_OBJECT_INVALID | non-existing object |
| RESULT_ARGUMENT_INVALID | MCU Firmware reports a unexpected/invalid command argument. |
| RESULT_FUNCTION_INVALID | Got a unknown result code from MCU Firmware. |
| RESULT_SERVICE_INVALID | service not supported by MCU Firmware |
| RESULT_COMM_TIMEOUT | Host to PSoC MCU timeout of spi communication. |
| RESULT_SEE_ERRNO | Inspect errno for further failure information. |

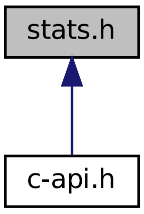
### 5.0.9 stats.h File Reference

Handle SPI communication statistics.

```
#include "resultcodes.h"
#include <stddef.h>
```
Include dependency graph for stats.h:



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



### Macros

- #define API_NAME "C-API"
- #define API_VERSION 9
- #define API_REVISION 13

### Functions

- unsigned int getCrcErrorCounter (enum RESULT ∗result=NULL)

  *Get count of checksum errors of SPI command detected by MCU.*
- unsigned int getResponseErrorCounter (enum RESULT ∗result=NULL)

  *Get count of the communication errors on SPI bus.*
- unsigned int getBusyCounter (enum RESULT ∗result=NULL)

  *Get count of detected busy states of MCU on SPI command.*
- unsigned int getTimeoutsCounter (enum RESULT ∗result=NULL)

  *Get count of SPI communication timeouts (after many attempts)*
- unsigned int getResetsCounter (enum RESULT ∗result=NULL)

  *Get count of MCU resets.*
- void resetStats (enum RESULT ∗result=NULL)

  *Reset error stats.*
- unsigned int getApiVersion (enum RESULT ∗result=NULL)

  *get software version of C-API library*
- unsigned int getPicVersion (enum RESULT ∗result=NULL)

  *get mcu firmware version*
- unsigned int getHwRevision (enum RESULT ∗result=NULL)

  *get hardware version of base board*

#### 5.0.9.1 Detailed Description

Handle SPI communication statistics.

In the SPI communication between the DHCOM and the MCU can be errors. Check on the stats if there are serious problems (hardware or software).

#### 5.0.9.2 Macro Definition Documentation

**API_NAME** `#define API_NAME "C-API"`

project name of 'C-API' library

**API_VERSION** `#define API_VERSION 9`

software version of C-API library

**API_REVISION** `#define API_REVISION 13`

software revision of C-API library

#### 5.0.9.3 Function Documentation

**getCrcErrorCounter()**
```
unsigned int getCrcErrorCounter (
        enum RESULT * result = NULL )
```

Get count of checksum errors of SPI command detected by MCU.

If the MCU detects a checksum error in the incoming command this counter is incremented. Not critical, but must not rise too fast.

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

Returns

count of crc errors

**getResponseErrorCounter()**
```
unsigned int getResponseErrorCounter (
        enum RESULT * result = NULL )
```

Get count of the communication errors on SPI bus.

Error count is not critical, but if the number rises too fast there may be a hardware problem with the SPI interface.

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

Returns

count of communication errors

**getBusyCounter()** `unsigned int getBusyCounter ( enum RESULT * result = NULL )`

Get count of detected busy states of MCU on SPI command.

The counter is incremented if the MCU is in the middle of a command processing and already the next order comes. May not rise quickly, otherwise there is a software problem (should be optimized).

**getTimeoutsCounter()** `unsigned int getTimeoutsCounter ( enum RESULT * result = NULL )`

Get count of SPI communication timeouts (after many attempts)

This counter is incremented when, after many attempts, the communication with the MCU is missing. Must be zero. If rising - there is a SW or HW problem.

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

Returns

count of SPI communication timeouts

**getResetsCounter()** `unsigned int getResetsCounter ( enum RESULT * result = NULL )`

Get count of MCU resets.

If the MCU no longer responds to periodic ping, DHCOM resets it. Must be zero and if it rises there is a problem with HW or MCU SW.

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

Returns

count MCU resets through DHCOM

**resetStats()** `void resetStats ( enum RESULT * result = NULL )`

Reset error stats.

Reset error counters (stats) of the SPI communication.

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

**getApiVersion()**
```
unsigned int getApiVersion (
        enum RESULT * result = NULL )
```

get software version of C-API library

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

Returns

integer with software version code.

**getPicVersion()**
```
unsigned int getPicVersion (
        enum RESULT * result = NULL )
```

get mcu firmware version

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

Returns

integer with software version code.

**getHwRevision()**
```
unsigned int getHwRevision (
        enum RESULT * result = NULL )
```

get hardware version of base board

Parameters

| | |
|---|---|
| *result* | pointer to pass RESULT of operation |

Returns

integer with hardware version of the base board.

# Index

~Bitmap
  Bitmap, 8

API_NAME
  stats.h, 30
API_REVISION
  stats.h, 31
API_VERSION
  stats.h, 31

Bitmap, 7
  ~Bitmap, 8
  Bitmap, 8
  data, 9
  height, 9
  pitch, 9
  size, 9
  width, 9
bitmap.h, 12
buffer
  FT_Bitmap_, 10
BUTTON
  buttons.h, 14
BUTTON_DN_LEFT
  buttons.h, 14
BUTTON_DN_RIGHT
  buttons.h, 14
BUTTON_MID
  buttons.h, 14
BUTTON_UP_LEFT
  buttons.h, 14
BUTTON_UP_RIGHT
  buttons.h, 14
buttons.h, 13
  BUTTON, 14
  BUTTON_DN_LEFT, 14
  BUTTON_DN_RIGHT, 14
  BUTTON_MID, 14
  BUTTON_UP_LEFT, 14
  BUTTON_UP_RIGHT, 14
  buttonsCallback, 14
  getButtonState, 15
  handleButtons, 14
  setButtonsCallback, 14
buttonsCallback
  buttons.h, 14

c-api.h, 15
closeApi
  init.h, 25

data
  Bitmap, 9
display.h, 16
  DISPLAY_DEFAULT_HEIGHT, 17
  DISPLAY_DEFAULT_WIDTH, 17
  displayBitmap, 21
  displayBitmap2, 21
  displayDrawRect, 20
  displayEnable, 18
  displayFill, 19
  displayFillRect, 19
  displayFlush, 22
  displayInvertRect, 20
  displayScreen, 21
  displaySetContrast, 18
  displaySetDimTimeout, 18
  displaySetOffTimeout, 19
  displaySwap, 22
  displayTakeScreenshot, 22
  displayWriteSplash, 22
DISPLAY_DEFAULT_HEIGHT
  display.h, 17
DISPLAY_DEFAULT_WIDTH
  display.h, 17
displayBitmap
  display.h, 21
displayBitmap2
  display.h, 21
displayDrawRect
  display.h, 20
displayEnable
  display.h, 18
displayFill
  display.h, 19
displayFillRect
  display.h, 19
displayFlush
  display.h, 22
displayInvertRect
  display.h, 20
displayScreen
  display.h, 21
displaySetContrast
  display.h, 18
displaySetDimTimeout
  display.h, 18
displaySetOffTimeout
  display.h, 19
displaySwap

display.h, 22
displayTakeScreenshot
display.h, 22
displayWriteSplash
display.h, 22

FT_Bitmap_, 9
buffer, 10
num_grays, 11
palette, 11
palette_mode, 11
pitch, 10
pixel_mode, 11
rows, 10
width, 10

getApiVersion
stats.h, 33
getBusyCounter
stats.h, 31
getButtonState
buttons.h, 15
getCrcErrorCounter
stats.h, 31
getHwRevision
stats.h, 33
getLedState
leds.h, 28
getPicVersion
stats.h, 33
getResetsCounter
stats.h, 32
getResponseErrorCounter
stats.h, 31
getResultCodeString
helpers.h, 23
getTimeoutsCounter
stats.h, 32

handleButtons
buttons.h, 14
height
Bitmap, 9
helpers.h, 23
getResultCodeString, 23

idleApi
init.h, 25
init.h, 24
closeApi, 25
idleApi, 25
openApi, 25

LED
leds.h, 27
LED_BUS
leds.h, 27
LED_ERR
leds.h, 27
LED_LAN
leds.h, 27
LED_RUN
leds.h, 27
leds.h, 26
getLedState, 28
LED, 27
LED_BUS, 27
LED_ERR, 27
LED_LAN, 27
LED_RUN, 27
writeLed, 27

num_grays
FT_Bitmap_, 11

openApi
init.h, 25

palette
FT_Bitmap_, 11
palette_mode
FT_Bitmap_, 11
pitch
Bitmap, 9
FT_Bitmap_, 10
pixel_mode
FT_Bitmap_, 11

resetStats
stats.h, 32
RESULT
resultcodes.h, 29
RESULT_API_NOT_OPEN
resultcodes.h, 29
RESULT_ARGUMENT_INVALID
resultcodes.h, 29
RESULT_COMM_TIMEOUT
resultcodes.h, 29
RESULT_FUNCTION_INVALID
resultcodes.h, 29
RESULT_OBJECT_INVALID
resultcodes.h, 29
RESULT_OK
resultcodes.h, 29
RESULT_SEE_ERRNO
resultcodes.h, 29
RESULT_SERVICE_INVALID
resultcodes.h, 29
resultcodes.h, 28
RESULT, 29
RESULT_API_NOT_OPEN, 29
RESULT_ARGUMENT_INVALID, 29
RESULT_COMM_TIMEOUT, 29
RESULT_FUNCTION_INVALID, 29
RESULT_OBJECT_INVALID, 29
RESULT_OK, 29
RESULT_SEE_ERRNO, 29
RESULT_SERVICE_INVALID, 29

rows
    FT_Bitmap_, 10

setButtonsCallback
    buttons.h, 14
size
    Bitmap, 9
stats.h, 29
    API_NAME, 30
    API_REVISION, 31
    API_VERSION, 31
    getApiVersion, 33
    getBusyCounter, 31
    getCrcErrorCounter, 31
    getHwRevision, 33
    getPicVersion, 33
    getResetsCounter, 32
    getResponseErrorCounter, 31
    getTimeoutsCounter, 32
    resetStats, 32

width
    Bitmap, 9
    FT_Bitmap_, 10
writeLed
    leds.h, 27