
# Simba Documentation

***Release master***

**Erik Moqvist**

**Oct 02, 2018**

# Contents

*Simba* is an Embedded Programming Platform. It aims to make embedded programming easy and portable.

Simba is written in C.

Project homepage: https://github.com/eerimoq/simba

CHAPTER 1

# Videos

Transmit CAN frames between a Nano32 and an Arduino Due. More videos are available on the *Videos* page.

## 1.1 Getting Started

### 1.1.1 Installation

There are three build systems available; *PlatformIO*, *Arduino IDE* and *Simba build system*. The *Simba build system* has more features than to the other two. It supports executing test suites, generating code coverage, profiling and more. Still, if you are familiar with *Arduino IDE* or *PlatformIO*, use that instead since it will be less troublesome.

**PlatformIO**

Install *Simba* in PlatformIO.

1. Install the PlatformIO IDE.
2. Start the *PlatformIO IDE* and open *PlatformIO -> Project Examples* and select *simba/blink*.
3. Click on *Upload* (the arrow image) in the top left corner.
4. The built-in LED blinks!
5. Done!

**Arduino IDE**

Install *Simba* in the Arduino IDE 1.6.10 as a third party board using the Boards Manager.

1. Open *File -> Preferences*.

2. Add these URL:s to *Additional Boards Manager URLs* (click on the icon to the right of the text field) and press *OK*.

```
https://raw.githubusercontent.com/eerimoq/simba-releases/master/arduino/avr/
↪package_simba_avr_index.json
https://raw.githubusercontent.com/eerimoq/simba-releases/master/arduino/sam/
↪package_simba_sam_index.json
https://raw.githubusercontent.com/eerimoq/simba-releases/master/arduino/esp/
↪package_simba_esp_index.json
https://raw.githubusercontent.com/eerimoq/simba-releases/master/arduino/esp32/
↪package_simba_esp32_index.json
```

3. Open *Tools* -> *Board: ...* -> *Boards Manager...* and type *simba* in the search box.
4. Click on *Simba by Erik Moqivst version x.y.z* and click *Install* and press *Close*.
5. Open *Tools* -> *Board: ...* -> *Boards Manager...* and select one of the Simba boards in the list.
6. Open *File* -> *Examples* -> *Simba* -> *blink*.
7. Verify and upload the sketch to your device.
8. The built-in LED blinks!
9. Done!


### Simba build system

The *Simba* development environment can be installed on *Linux (Ubuntu 14)*.

1. Execute the one-liner below to install *Simba*.

```
$ mkdir simba && \
  cd simba && \
  sudo apt install ckermit valgrind cppcheck cloc python python-pip doxygen git␣
↪lcov && \
  sudo apt install avrdude gcc-avr binutils-avr gdb-avr avr-libc && \
  sudo apt install bossa-cli gcc-arm-none-eabi && \
  sudo apt install make unrar autoconf automake libtool gcc g++ gperf \
                       flex bison texinfo gawk ncurses-dev libexpat-dev \
                       python-serial sed libtool-bin pmccabe help2man \
                       python-pyelftools unzip && \
  sudo pip install pyserial xpect readchar sphinx breathe sphinx_rtd_theme && \
  (git clone --recursive https://github.com/pfalcon/esp-open-sdk && \
   cd esp-open-sdk && \
   make) && \
  wget https://github.com/eerimoq/simba-releases/raw/master/arduino/esp32/tools/
↪xtensa-esp32-elf-linux$(getconf LONG_BIT)-1.22.0-59.tar.gz && \
  tar xf xtensa-esp32-elf-linux$(getconf LONG_BIT)-1.22.0-59.tar.gz && \
  rm xtensa-esp32-elf-linux$(getconf LONG_BIT)-1.22.0-59.tar.gz && \
  git clone --recursive https://github.com/eerimoq/simba
```

2. Setup the environment.

```
$ cd simba
$ source setup.sh
```

2. Build and upload the blink example to your device. Replace `<my-serial-port>` with your serial port name.

```
$ cd examples/blink
$ make -s BOARD=nano32 SERIAL_PORT=<my-serial-port> upload
```

3. The built-in LED blinks!
4. Done!

---

**Note:** Some boards, such as the *SPC56D Discovery*, require a specific toolchain to build. Such cases are documented on the individual board documentation page.

---

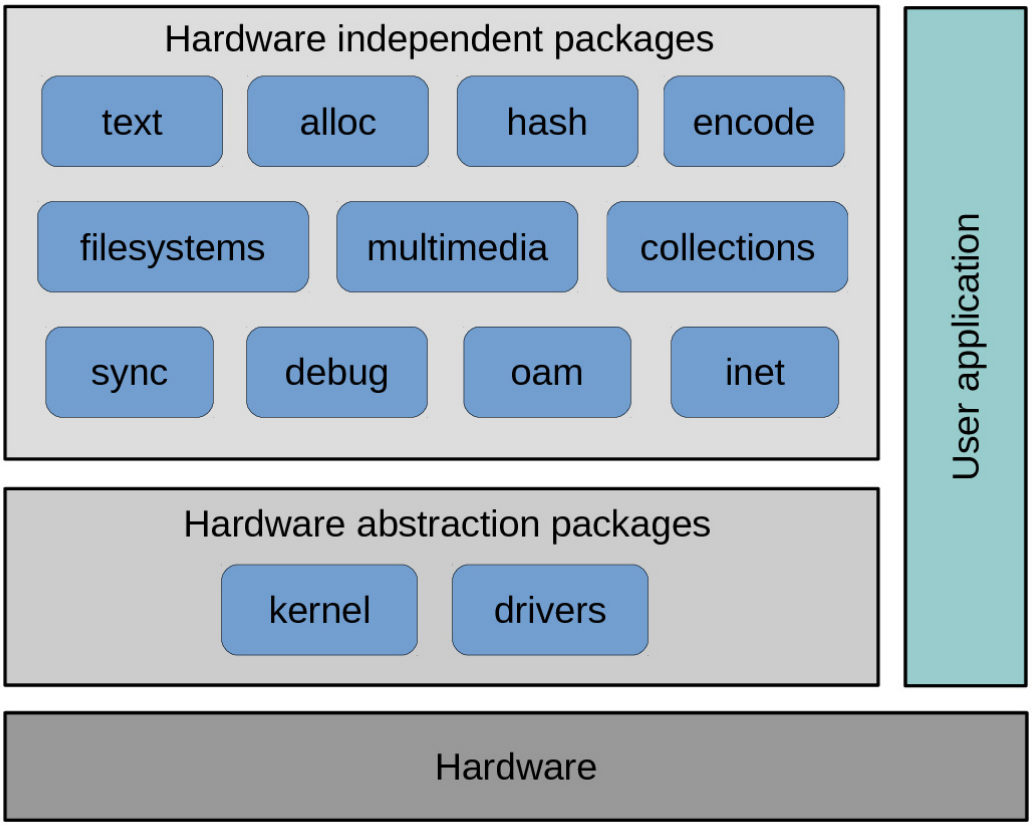
## 1.2 User Guide

This guide is intended for users of the Simba Embedded Programming Platform and the *Simba build system*. Parts of the guide is applicable to other build systems as well, in particular the configuration section.

The Simba installation guide can be found on the *Getting Started* page.

### 1.2.1 Software architecture

Below is a picture of all packages and their relation to the hardware. At the bottom is the hardware. On top of the hardware is the kernel and drivers packages, which exports a hardware independent interface that other packages and the user application can use. The user application on the right can use any package, and in rare cases directly access the hardware registers.




**Contents:**

### Environment setup

The first step is always to setup the *Simba* environment. It's a simple matter of sourcing a setup-script in the simba root folder.

This step only applies to the *Simba build system*, and not to the *Arduino IDE* or *PlatformIO*.

```
$ cd simba/simba
$ source setup.sh
```

### Hello World application

Let's start with the *Simba* "Hello World" application. It examplifies what an application is and how to build and run it.

It consists of two files; `main.c` and `Makefile`.

### main.c

main.c defines the application entry function `main()`.

```
#include "simba.h"

int main()
```

(continues on next page)

(continued from previous page)

```
{
    /* Initialize modules and start the scheduler. */
    sys_start();

    std_printf(FSTR("Hello world!\n"));

    return (0);
}
```

### Makefile

Makefile contains build configuration of the application.

```
NAME = hello_world
BOARD ?= linux

RUN_END_PATTERN = "Hello world!"
RUN_END_PATTERN_SUCCESS = "Hello world!"

SIMBA_ROOT = ../..
include $(SIMBA_ROOT)/make/app.mk
```

### Build and run

Compile, link and run it by typing the commands below in a shell:

```
$ cd examples/hello_world
$ make -s run
<build system output>
Hello world!
$
```

Cross-compile, link and then run on an Arduino Due:

```
$ cd examples/hello_world
$ make -s BOARD=arduino_due run
<build system output>
Hello world!
$
```

### Applications, packages and modules

*Simba* has three software components; the application, the package and the module.

### Application

An application is an executable consisting of zero or more packages.

```
myapp
├── main.c
└── Makefile
```

### Development workflow

Build and run often! More to be added, hopefully.

### Package

A package is a container of modules.

A package file tree **must** be organized as seen below. This is required by the build framework and *Simba* tools.

See the inline comments for details about the files and folders contents.

```
mypkg
├── mypkg
│   ├── doc                    # package documentation
│   ├── __init__.py
│   ├── src                    # package source code
│   │   ├── mypkg
│   │   │   ├── module1.c
│   │   │   └── module1.h
│   │   ├── mypkg.h            # package header file
│   │   └── mypkg.mk           # package makefile
│   └── tst                    # package test code
│       └── module1
│           ├── main.c
│           └── Makefile
└── setup.py
```

### Development workflow

The package development workflow is fairly straight forward. Suppose we want to add a new module to the file tree above. Create `src/mypkg/module2.h` and `src/mypkg/module2.c`, then include `mypkg/module2.h` in `src/mypkg.h` and add `mypkg/module2.c` to the list of source files in `src/mypkg.mk`. Create a test suite for the module. It consists of the two files `tst/module2/main.c` and `tst/module2/Makefile`.

It's often conveniant to use an existing modules' files as skeleton for the new module.

After adding the module `module2` the file tree looks like this.

```
mypkg
├── mypkg
│   ├── doc
│   ├── __init__.py
│   ├── src
│   │   ├── mypkg
│   │   │   ├── module1.c
│   │   │   ├── module1.h
│   │   │   ├── module2.c
│   │   │   └── module2.h
│   │   ├── mypkg.h
│   │   └── mypkg.mk
│   └── tst
│       ├── module1
│       │   ├── main.c
│       │   └── Makefile
```

(continues on next page)

(continued from previous page)

```
│       └── module2
│           ├── main.c
│           └── Makefile
└── setup.py
```

Now, build and run the test suite to make sure the empty module implementation compiles and can be executed.

```
$ cd tst/module2
$ make -s run
```

Often the module development is started by implementing the module header file and at the same time write test cases. Test cases are not only useful to make sure the implementation works, but also to see how the module is intended to be used. The module interface becomes cleaner and easier to use it you actually start to use it yourself by writing test cases! All users of your module will benefit from this!

So, now we have an interface and a test suite. It's time to start the implementation of the module. Usually you write some code, then run the test suite, then fix the code, then run the tests again, then you realize the interface is bad, change it, change the implementation, change the test, change, change... and so it goes on until you are satisfied with the module.

Try to update the comments and documentation during the development process so you don't have to do it all in the end. It's actually quite useful for yourself to have comments. You know, you forget how to use your module too!

The documentation generation framework uses doxygen, breathe and sphinx. That means, all comments in the source code should be written for doxygen. Breathe takes the doxygen output as input and creates input for sphinx. Sphinx then generates the html documentation.

Just run `make` in the `doc` folder to generate the html documentation.

```
$ cd doc
$ make
$ firefox _build/html/index.html    # open the docs in firefox
```

### Namespaces

All exported symbols in a package must have the prefix `<package>_<module>_`. This is needed to avoid namespace clashes between modules with the same name in different packages.

There cannot be two packages with the same name, for the namespace reason. All packages must have unique names! There is one exception though, the three *Simba* packages; kernel, drivers and slib. Those packages does *not* have the package name as prefix on exported symbols.

```
int mypackage_module1_foo(void);

int mypackage_module2_bar(void);
```

### Module

A module is normally a header and a source file.

### Configuration

### Standard Library

The *Library Reference* is configured at compile time using defines named `CONFIG_*`. The default configuration includes most functionality, as most application wants that. If an application has special requirements, for example memory constraints, it has to be configured to remove unnecessaray functionality.

### Search order

Highest priority first.

### Simba build system

1. Command line as `CDEFS_EXTRA="<configuration variable>=<value>"`.
2. A file named `config.h` in the application root folder.
3. The default configuration file, src/config_default.h.

### PlatformIO

1. The variable `build_flags` in `platformio.ini` as `build_flags = -D<configuration variable>=<value>`.
2. A file named `config.h` in the application source folder `src`.
3. The default configuration file, src/config_default.h.

### Arduino IDE

1. A file (also called a *tab*) named `config.h` in the sketch.
2. The default configuration file, src/config_default.h.

### Default Settings

In this section you can find some important and widly used settings.

### Console UART

The default configuration of the system console's UART can be found at the page of particular boards in the *Boards* in the Default Configuration section.

### Variables

All configuration variables are listed below. Their default values are defined in src/config_default.h.

### Defines

**PORT_HAS_I2C_SOFT**

**PORT_HAS_OWI**

**PORT_HAS_PIN**

**PORT_HAS_UART**

**PORT_HAS_XBEE**

**PORT_HAS_XBEE_CLIENT**

**PORT_HAS_HX711**

**PORT_HAS_GNSS**

**PORT_HAS_HD44780**

**PORT_HAS_ICSP_SOFT**

**PORT_HAS_JTAG_SOFT**

**PORT_HAS_I2C**
Used to include driver header files and the c-file source.

**CONFIG_SOFTWARE_I2C**

**PORT_HAS_DS3231**

**PORT_HAS_EEPROM_I2C**

**PORT_HAS_SHT3XD**

**PORT_HAS_DS18B20**

**PORT_HAS_BMP280**

**CONFIG_SYS_CONFIG_STRING**
The system configuration string contains a list of all configration varialbes and their values.

**CONFIG_SYS_SIMBA_MAIN_STACK_MAX**
Main thread stack size for ports with a fixed size main thread stack.

**CONFIG_SYS_PANIC_KICK_WATCHDOG**
Kick the watchdog in `sys_panic()` before writing to the console.

**CONFIG_SYS_PANIC_BACKTRACE_DEPTH**
Maximum backtrace depth in `sys_panic()`.

**CONFIG_ASSERT**
Assertions are used to check various conditions during the application execution. A typical usage is to validate function input arguments.

**CONFIG_ASSERT_FORCE_FATAL**
Force all assertions to be fatal.

**CONFIG_ASSERT_FORCE_PANIC**
Force all assertions to be panics.

**CONFIG_FATAL_ASSERT**
Enable fatal assertions, `FATAL_ASSERT*()`.

**CONFIG_PANIC_ASSERT**
Enable panic assertions, `PANIC_ASSERT*()`.

**CONFIG_PANIC_ASSERT_FILE_LINE**
Include file and line in panic assert output.

**CONFIG_DEBUG**
Include more debug information.

**CONFIG_LINUX_SOCKET_DEVICE**
Enable linux driver implementations as TCP sockets. Can be used to simulate driver communication in an application running on linux.

**CONFIG_ADC**
Enable the adc driver.

**CONFIG_ANALOG_INPUT_PIN**
Enable the analog_input_pin driver.

**CONFIG_ANALOG_OUTPUT_PIN**
Enable the analog_output_pin driver.

**CONFIG_POWER**
Enable the power driver.

**CONFIG_CAN**
Enable the can driver.

**CONFIG_CAN_FRAME_TIMESTAMP**
Timestamp received CAN frames.

**CONFIG_CHIPID**
Enable the chipid driver.

**CONFIG_RANDOM**
Enable the random driver.

**CONFIG_WS2812**
Enable the ws2812 driver.

**CONFIG_LED_7SEG_HT16K33**
Enable the led_7seg_ht16k33 driver.

**CONFIG_SHT3XD**
Enable the sht3xd driver.

**CONFIG_DAC**
Enable the dac driver.

**CONFIG_DS18B20**
Enable the ds18b20 driver.

**CONFIG_DS3231**
Enable the ds3231 driver.

**CONFIG_EEPROM_SOFT**
Enable the eeprom_soft driver.

**CONFIG_ESP_WIFI**
Enable the esp_wifi driver.

**CONFIG_ESP_WIFI_FS_COMMAND_STATUS**
Debug file system command to print the Espressif WiFi status.

**CONFIG_EXTI**
Enable the exti driver.

**CONFIG_PCINT**
Enable the pin change interrupt driver.

**CONFIG_FLASH**
Enable the flash driver.

**CONFIG_I2C**
Enable the i2c driver.

**CONFIG_I2C_SOFT**
Enable the i2c_soft driver.

**CONFIG_ICSP_SOFT**
Enable the icsp_soft driver.

**CONFIG_JTAG_SOFT**
Enable the jtag_soft driver.

**CONFIG_EEPROM_I2C**
Enable the eeprom_i2c driver.

**CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS**
Number of write retry attemps before giving up accessing a i2c eeprom.

**CONFIG_MCP2515**
Enable the mcp2515 driver.

**CONFIG_NRF24L01**
Enable the nrf24l01 driver.

**CONFIG_OWI**
Enable the owi driver.

**CONFIG_PIN**
Enable the pin driver.

**CONFIG_PIN_FS_COMMAND_READ**
Debug file system command to read the current value of a pin.

**CONFIG_PIN_FS_COMMAND_SET_MODE**
Debug file system command to set the mode of a pin.

**CONFIG_PIN_FS_COMMAND_WRITE**
Debug file system command to write a value to a pin.

**CONFIG_PWM**
Enable the pwm driver.

**CONFIG_PWM_SOFT**
Enable the pwm_soft driver.

**CONFIG_SD**
Enable the sd driver.

**CONFIG_SPI**
Enable the spi driver.

**CONFIG_UART**
Enable the uart driver.

**CONFIG_UART_FS_COUNTERS**
Debug file system uart counters.

**CONFIG_UART_SOFT**
Enable the uart_soft driver.

**CONFIG_USB**
Enable the usb driver.

**CONFIG_USB_DEVICE**
Enable the usb_device driver.

**CONFIG_USB_DEVICE_FS_COMMAND_LIST**
Debug file system command to list all USB devices.

**CONFIG_USB_HOST**
Enable the usb_host driver.

**CONFIG_USB_HOST_FS_COMMAND_LIST**
Debug file system command to list all USB devices connected to the USB host.

**CONFIG_WATCHDOG**
Enable the watchdog driver.

**CONFIG_XBEE**
Enable the xbee driver.

**CONFIG_XBEE_DATA_MAX**
Maximum xbee driver frame data size.

**CONFIG_XBEE_CLIENT**
Enable the xbee_client driver.

**CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK**
xbee_client driver debug log mask.

**CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS**
xbee_client packet response timeout in milliseconds.

**CONFIG_HX711**
Enable the hx711 driver.

**CONFIG_GNSS**
Enable the gnss driver.

**CONFIG_GNSS_DEBUG_LOG_MASK**
GNSS driver debug log mask.

**CONFIG_BMP280**
Enable the bmp280 driver.

**CONFIG_BMP280_COVERTION_TIMEOUT_MS**
BMP280 convertion timeout in milliseconds.

**CONFIG_BMP280_DEBUG_LOG_MASK**
BMP280 driver debug log mask.

**CONFIG_DHT**
Enable the dht driver.

**CONFIG_MODULE_INIT_RWLOCK**
Initialize the module at system startup.

**CONFIG_MODULE_INIT_FS**
Initialize the fs module at system startup.

**CONFIG_MODULE_INIT_SETTINGS**
Initialize the settings module at system startup.

**CONFIG_MODULE_INIT_STD**
Initialize the std module at system startup.

**CONFIG_MODULE_INIT_SEM**
Initialize the sem module at system startup.

**CONFIG_MODULE_INIT_TIMER**
Initialize the timer module at system startup.

**CONFIG_MODULE_INIT_LOG**
Initialize the log module at system startup.

**CONFIG_MODULE_INIT_CHAN**
Initialize the chan module at system startup.

**CONFIG_MODULE_INIT_THRD**
Initialize the thrd module at system startup.

**CONFIG_MODULE_INIT_ADC**
Initialize the adc driver module at system startup.

**CONFIG_MODULE_INIT_ANALOG_INPUT_PIN**
Initialize the analog_input_pin driver module at system startup.

**CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN**
Initialize the analog_output_pin driver module at system startup.

**CONFIG_MODULE_INIT_CAN**
Initialize the can driver module at system startup.

**CONFIG_MODULE_INIT_CHIPID**
Initialize the chipid driver module at system startup.

**CONFIG_MODULE_INIT_RANDOM**
Initialize the random driver module at system startup.

**CONFIG_MODULE_INIT_DAC**
Initialize the dac driver module at system startup.

**CONFIG_MODULE_INIT_DS18B20**
Initialize the ds18b20 driver module at system startup.

**CONFIG_MODULE_INIT_DHT**
Initialize the dht driver module at system startup.

**CONFIG_DS18B20_FS_COMMAND_LIST**
Debug file system command to list all DS18B20 sensors on the bus.

**CONFIG_DHT_COMMAND_READ**
Debug file system command to read DHT sensor on the bus.

**CONFIG_MODULE_INIT_DS3231**
Initialize the ds3231 driver module at system startup.

**CONFIG_MODULE_INIT_ESP_WIFI**
Initialize the esp_wifi driver module at system startup.

**CONFIG_MODULE_INIT_EXTI**
Initialize the exti driver module at system startup.

**CONFIG_MODULE_INIT_FLASH**
Initialize the flash driver module at system startup.

**CONFIG_MODULE_INIT_I2C**
Initialize the i2c driver module at system startup.

**CONFIG_MODULE_INIT_I2C_SOFT**
Initialize the i2c_soft driver module at system startup.

**CONFIG_MODULE_INIT_MCP2515**
Initialize the mcp2515 driver module at system startup.

**CONFIG_MODULE_INIT_NRF24L01**
Initialize the nrf24l01 driver module at system startup.

**CONFIG_MODULE_INIT_OWI**
Initialize the owi driver module at system startup.

**CONFIG_MODULE_INIT_PIN**
Initialize the pin driver module at system startup.

**CONFIG_MODULE_INIT_POWER**
Initialize the power driver module at system startup.

**CONFIG_MODULE_INIT_PWM**
Initialize the pwm driver module at system startup.

**CONFIG_MODULE_INIT_PWM_SOFT**
Initialize the pwm_soft driver module at system startup.

**CONFIG_MODULE_INIT_SD**
Initialize the sd driver module at system startup.

**CONFIG_MODULE_INIT_SPI**
Initialize the spi driver module at system startup.

**CONFIG_MODULE_INIT_UART**
Initialize the uart driver module at system startup.

**CONFIG_MODULE_INIT_UART_SOFT**
Initialize the uart_soft driver module at system startup.

**CONFIG_MODULE_INIT_USB**
Initialize the usb driver module at system startup.

**CONFIG_MODULE_INIT_USB_DEVICE**
Initialize the usb_device driver module at system startup.

**CONFIG_MODULE_INIT_USB_HOST**
Initialize the usb_host driver module at system startup.

**CONFIG_MODULE_INIT_WATCHDOG**
Initialize the watchdog driver module at system startup.

**CONFIG_MODULE_INIT_BUS**
Initialize the bus module at system startup.

**CONFIG_MODULE_INIT_INET**
Initialize the inet module at system startup.

**CONFIG_MODULE_INIT_PING**
Initialize the ping module at system startup.

**CONFIG_MODULE_INIT_SOCKET**
Initialize the socket module at system startup.

**CONFIG_MODULE_INIT_NETWORK_INTERFACE**
Initialize the network_interface module at system startup.

**CONFIG_MODULE_INIT_SSL**
Initialize the ssl module at system startup.

**CONFIG_MODULE_INIT_UPGRADE**
Initialize the upgrade module at system startup.

**CONFIG_MODULE_INIT_RE**
Initialize the regular expression module at system startup.

**CONFIG_I2C_FS_COMMAND_READ**
Debug file system command to read from a i2c bus.

**CONFIG_I2C_FS_COMMAND_WRITE**
Debug file system command to write to a i2c bus.

**CONFIG_I2C_FS_COMMAND_SCAN**
Debug file system command to scan a i2c bus.

**CONFIG_LOG_FS_COMMANDS**
Debug file system command to list all log objects.

**CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST**
Debug file system command to list all network interfaces.

**CONFIG_PING_FS_COMMAND_PING**
Debug file system command to ping a host.

**CONFIG_SERVICE_FS_COMMAND_LIST**
Debug file system command to list all services.

**CONFIG_SERVICE_FS_COMMAND_START**
Debug file system command to start a service.

**CONFIG_SERVICE_FS_COMMAND_STOP**
Debug file system command to stop a services.

**CONFIG_SETTINGS_FS_COMMAND_LIST**
Debug file system command to list all settings.

**CONFIG_SETTINGS_FS_COMMAND_READ**
Debug file system command to read the value of a setting.

**CONFIG_SETTINGS_FS_COMMAND_RESET**
Debug file system command to reset the settings to their original values.

**CONFIG_SETTINGS_FS_COMMAND_WRITE**
Debug file system command to write a value to a setting.

**CONFIG_SYS_FS_COMMANDS**
Sys module debug file system commands.

**CONFIG_THRD_FS_COMMANDS**
Thrd module debug file system commands.

**CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER**
Debug file system command to enter the application.

**CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE**
Debug file system command to erase the application.

**CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID**
Debug file system command to check if the application is valid.

**CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER**
Debug file system command to enter the bootloader.

**CONFIG_FS_PATH_MAX**
The maximum length of an absolute path in the file system.

**CONFIG_FS_COMMAND_ARGS_MAX**
Maximum number of command arguments, including the command name.

**CONFIG_FS_FS_COMMAND_APPEND**
Debug file system command to append to a file.

**CONFIG_FS_FS_COMMAND_COUNTERS_LIST**
Debug file system command to list all counters.

**CONFIG_FS_FS_COMMAND_COUNTERS_RESET**
Debug file system command to set all counters to zero.

**CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST**
Debug file system command to list all registered file systems.

**CONFIG_FS_FS_COMMAND_LIST**
Debug file system command to list all registered file systems.

**CONFIG_FS_FS_COMMAND_FORMAT**
Debug file system command to format a file system.

**CONFIG_FS_FS_COMMAND_PARAMETERS_LIST**
Debug file system command to list all parameters.

**CONFIG_FS_FS_COMMAND_READ**
Debug file system command to read from a file.

**CONFIG_FS_FS_COMMAND_REMOVE**
Debug file system command to remove a file.

**CONFIG_FS_FS_COMMAND_WRITE**
Debug file system command to write to a file.

**CONFIG_MONITOR_THREAD**
Start the monitor thread to gather statistics of the scheulder.

**CONFIG_MONITOR_THREAD_PERIOD_US**
Default period of the monitor thread in microseconds.

**CONFIG_PREEMPTIVE_SCHEDULER**
Use a preemptive scheduler.

**CONFIG_PROFILE_STACK**
Profile the stack usage in runtime. It's a cheap operation and is recommended to have enabled.

**CONFIG_SETTINGS_AREA_SIZE**
Size of the settings area. This size *MUST* have the same size as the settings generated by the settings.py script.

**CONFIG_SETTINGS_BLOB**
Enable the blob setting type.

**CONFIG_SHELL_COMMAND_MAX**
Maximum number of characters in a shell command.

**CONFIG_SHELL_HISTORY_SIZE**
Size of the shell history buffer.

**CONFIG_SHELL_MINIMAL**
Minimal shell functionality to minimize the code size of the shell module.

**CONFIG_SHELL_PROMPT**
The shell prompt string.

**CONFIG_SOCKET_RAW**
Raw socket support.

**CONFIG_SPIFFS**
SPIFFS is a flash file system applicable for boards that has a reasonably big modifiable flash.

**CONFIG_FAT16**
FAT16 is a file system.

**CONFIG_FILESYSTEM_GENERIC**
Generic file system.

**CONFIG_START_CONSOLE**
Start the console device (UART/USB CDC) on system startup.

**CONFIG_START_CONSOLE_DEVICE_INDEX**
Console device index.

**CONFIG_START_CONSOLE_UART_BAUDRATE**
Console UART baudrate.

**CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE**
Console UART reception buffer size.

**CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE**
Console USB CDC control interface number.

**CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN**
Console USB CDC input endpoint.

**CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT**
Console USB CDC output endpoint.

**CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION**
Wait for the host to connect after starting the console.

**CONFIG_START_FILESYSTEM**
Configure a default file system.

**CONFIG_START_FILESYSTEM_FAT16**
Start a FAT16 file system.

**CONFIG_START_FILESYSTEM_SPIFFS**
Start a flash file system.

**CONFIG_START_FILESYSTEM_ADDRESS**
Configure a default file system start address.

**CONFIG_START_FILESYSTEM_SIZE**
Configure a default file system size.

**CONFIG_START_NVM**
Configure a default non-volatile memory.

**CONFIG_NVM_SIZE**
Non-volatile memory size in bytes.

**CONFIG_NVM_EEPROM_SOFT**
Use the software EEPROM implementation in the non-volatile memory module.

**CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE**
Non-volatile memory software EEPROM block 0 size. Must be a multiple of `CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE`.

**CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE**
Non-volatile memory software EEPROM block 1 size. Must be a multiple of `CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE`.

**CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE**
Non-volatile software EEPROM chunk size. Must be a power of two.

**CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX**
Non-volatile software EEPROM flash device index.

**CONFIG_NVM_FS_COMMAND_READ**
Debug file system command to read for non-volatile memory.

**CONFIG_NVM_FS_COMMAND_WRITE**
Debug file system command to write for non-volatile memory.

**CONFIG_START_NETWORK**
Setup the ip stack and connect to all configured networks.

**CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT**
WiFi connect timeout is seconds.

**CONFIG_START_NETWORK_INTERFACE_WIFI_SSID**
SSID of the WiFi to connect to.

**CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD**
Password of the WiFi to connect to.

**CONFIG_START_SHELL**
Start a shell thread communication over the console channels.

**CONFIG_START_SHELL_PRIO**
Shell thread priority.

**CONFIG_START_SHELL_STACK_SIZE**
Shell thread stack size in words.

**CONFIG_START_SOAM**
Start a SOAM thread communication over the console channels.

**CONFIG_START_SOAM_PRIO**
SOAM thread priority.

**CONFIG_START_SOAM_STACK_SIZE**
SOAM thread stack size in words.

**CONFIG_START_SOAM_RX_BUFFER_SIZE**
SOAM reception buffer size.

**CONFIG_START_SOAM_TX_BUFFER_SIZE**
SOAM transmission buffer size.

**CONFIG_STD_OUTPUT_BUFFER_MAX**
Maximum number of bytes in the print output buffer.

**CONFIG_FLOAT**
Use floating point numbers instead of intergers where applicable.

**CONFIG_ALIGNMENT**
Memory alignment for runtime memory allocations.

**CONFIG_SYSTEM_TICK_FREQUENCY**
System tick frequency in Hertz.

**CONFIG_SYSTEM_INTERRUPTS**
Use interrupts.

**CONFIG_SYSTEM_INTERRUPT_STACK_SIZE**
Interrupt stack size in bytes. Set to a value greater than zero to enable the interrupt stack.

**CONFIG_THRD_CPU_USAGE**
Calculate thread CPU usage.

**CONFIG_THRD_DEFAULT_LOG_MASK**
Default thread log mask.

**CONFIG_RE_DEBUG_LOG_MASK**
Regular expression module debug log mask.

**CONFIG_THRD_ENV**
Each thread has a list of environment variables associated with it. A typical example of an environment variable is "CWD" - Current Working Directory.

**CONFIG_THRD_IDLE_STACK_SIZE**
Stack size of the idle thread.

**CONFIG_THRD_MONITOR_STACK_SIZE**
Stack size of the monitor thread.

**CONFIG_THRD_SCHEDULED**
Count the number of times each thread has been scheduled.

**CONFIG_THRD_STACK_HEAP**
Enable the thread stack heap allocator.

**CONFIG_THRD_STACK_HEAP_SIZE**
Size in bytes of the thread stack heap.

**CONFIG_THRD_TERMINATE**
Threads are allowed to terminate.

**CONFIG_USB_DEVICE_VID**
USB device vendor id.

**CONFIG_USB_DEVICE_PID**
USB device product id.

**CONFIG_EMACS_COLUMNS_MAX**
Number of colums in Emacs text editor.

**CONFIG_EMACS_ROWS_MAX**
Number of rows in Emacs text editor.

**CONFIG_EMACS_HEAP_SIZE**
Heap size of the emacs text editor.

**CONFIG_SYSTEM_TICK_SOFTWARE**
System tick using a software timer instead of a hardware timer. Suitable for ESP8266 to enable software PWM.

**CONFIG_HTTP_SERVER_SSL**
Add support to wrap the HTTP server in SSL, creating a HTTPS server.

**CONFIG_HARNESS_SLEEP_MS**
Sleep in the test harness before executing the first testcase.

**CONFIG_HARNESS_EXPECT_BUFFER_SIZE**
Maximum buffer size the expect function can handle.

**CONFIG_HARNESS_BACKTRACE_DEPTH_MAX**
The maximum harness backtrace depth.

**CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX**
The maximum harness write backtrace depth.

**CONFIG_HARNESS_MOCK_ENTRIES_MAX**
Maximum number of mock entries that can be allocated at any given moment, required for harness_mock_() functions.

**CONFIG_HARNESS_EARLY_EXIT**
Call sys_exit() immediately on test failure to stop the test suite execution as early as possible.

**CONFIG_HARNESS_DEBUG**
Output debug information from the harness.

**CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE**
Size of the HTTP server request buffer. This buffer is used when parsing received HTTP request headers.

**CONFIG_CRC_TABLE_LOOKUP**
Use lookup tables for CRC calculations. It is faster, but uses more memory.

**CONFIG_SPC5_BOOT_ENTRY_RCHW**

**CONFIG_SPC5_RAM_CLEAR_ALL**

**CONFIG_SPC5_RELOCATE_INIT**

**CONFIG_SPC5_WATCHDOG_DISABLE**

**CONFIG_TIME_UNIX_TIME_TO_DATE**
Include the function time_unix_time_to_date().

**CONFIG_SOAM_EMBEDDED_DATABASE**
Embed the SOAM database in the application.

**CONFIG_SYS_LOG_MASK**
System module log mask.

**CONFIG_SYS_RESET_CAUSE**
Save reset cause at startup.

**CONFIG_SYS_MEASURE_INTERRUPT_LOAD**
Save reset cause at startup.

**CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ**
The external oscillator frequency in Hertz.

**CONFIG_FLASH_DEVICE_SEMAPHORE**
Semaphore protected device access in the flash driver module.

**CONFIG_EEPROM_SOFT_SEMAPHORE**
Semaphore protected software eeprom accesses.

**CONFIG_EEPROM_SOFT_CRC_32**

**CONFIG_EEPROM_SOFT_CRC_CCITT**

**CONFIG_EEPROM_SOFT_CRC**
Software eeprom crc algorithm.

**CONFIG_EEPROM_SOFT_OVERWRITE_IDENTICAL_DATA**
Overwrite identical data in software eeprom.

### lwIP

Use `config.h` to fully configure lwIP and all of its modules. You do not need to define every option that lwIP provides; if you do not define an option, a default value will be used. Therefore, your `config.h` provides a way to override much of the behavior of lwIP.

By default *Simba* overrides a few of the variables in src/inet/lwipopts.h.

### Module support (Code size)

### Enabling and disabling modules

You can tune your code size by only compiling the features you really need. The following is a list of what gets compiled in "out of the box" with lwIP.

Default inclusions:

- ARP (`LWIP_ARP`)
- IP and fragmentation (`IP_FRAG`) and reassembly (`IP_REASSEMBLY`)
- Raw IP PCB support (`LWIP_RAW`)
- UDP (`LWIP_UDP`) and UDP-Lite (`LWIP_UDPLITE`)
- TCP (`LWIP_TCP`) – this is a big one!
- Statistics (`LWIP_STATS`)

Default exclusions:

- DHCP (`LWIP_DHCP`)
- AUTOIP (`LWIP_AUTOIP`)
- SNMP (`LWIP_SNMP`)
- IGMP (`LWIP_IGMP`)
- PPP (`PPP_SUPPORT`)

If you would like to change this, then you just need to set the options listed below. For example, if you would like to disable UDP and enable DHCP, the following `config.h` file would do it:

```
/* Disable UDP */
#define LWIP_UDP 0

/* Enable DHCP */
#define LWIP_DHCP 1
```

### Memory management (RAM usage)

### Memory pools

In an embedded environment, memory pools make for fast and efficient memory allocation. lwIP provides a flexible way to manage memory pool sizes and organization.

lwIP reserves a fixed-size static chunk of memory in the data segment, which is subdivided into the various pools that lwip uses for the various data structures. For example, there is a pool just for struct tcp_pcb's, and another pool just for struct udp_pcb's. Each pool can be configured to hold a fixed number of data structures; this number can be changed in the `config.h` file by changing the various `MEMP_NUM_*` values. For example, `MEMP_NUM_TCP_PCB` and `MEMP_NUM_UDP_PCB` control the maximum number of tcp_pcb and udp_pcb structures that can be active in the system at any given time.

It is also possible to create custom memory pools in addition to the standard ones provided by lwIP.

### Dynamic allocation: mem_malloc

lwIP uses a custom function mem_malloc for all dynamic allocation; therefore, it is easy to change how lwIP uses its RAM. There are three possibilities provided out-of-the-box:

1. (default) lwIP's custom heap-based mem_malloc. By default, lwIP uses a statically-allocated chunk of memory like a heap for all memory operations. Use `MEM_SIZE` to change the size of the lwIP heap.
2. C standard library malloc and free. If you wish to have lwIP use the standard library functions provided by your compiler/architecture, then define the option `MEM_LIBC_MALLOC`.
3. Memory pools. lwIP can also emulate dynamic allocation using custom memory pools (see that chapter for more information). This involves the options `MEM_USE_POOLS` and `MEMP_USE_CUSTOM_POOLS` and a new custom file `lwippools.h`.

### Understanding/changing memory usage

lwIP uses memory for:

- code (depending on your system, may use ROM instead of RAM)
- statically allocated variables (some initialized, some not initialized)
- task stack
- dynamically allocated memory
  - heap
  - memp pools

Unless you use a C library heap implementation (by defining `MEM_LIBC_MALLOC` to 1), dynamically allocated memory must be statically allocated somewhere. This means you reserve a specific amount of memory for the heap or the memp pools from which the code dynamically allocates memory at runtime.

The size of this heap and memp pools can be adjusted to save RAM:

There are 3 types of pbufs:

- REF/ROM, RAM and POOL. `PBUF_POOL_SIZE` * `PBUF_POOL_BUFSIZE` only refers to type POOL.
- RAM pbufs are allocated in the memory defined by `MEM_SIZE` (this memory is not used much aside from RAM pbufs) - this is the *heap* and it is allocated as mem_memory.

- REF/ROM pbufs as well as pcbs and some other stuff is allocated from dedicated pools per structure type. The amount of structures is defined by the various MEMP_NUM_ defines. Together, this memory is allocated as memp_memory and it *includes* the pbuf POOL.

However, if you define MEMP_MEM_MALLOC to 1 in your config.h, *every* piece of dynamically allocated memory will come from the heap (the size of which is defined by MEM_SIZE). If you then even define MEM_LIBC_MALLOC to 1, too, lwIP doesn't need extra memory for dynamically allocated memory but only uses the C library heap instead. However, you then have to make sure that this heap is big enough to run your application.

To tweak the various MEMP_NUM_ defines, define LWIP_STATS=1 and LWIP_STATS_DISPLAY=1 and call *stats_display()* to see how many entries of each pool are used (or have a look at the global variable lwip_stats instead).

#### Fine-tuning even more

To see the options that you can set, open 3pp/lwip-1.4.1/src/include/lwip/opt.h. This file is fully commented and explains how many of the options are used.

### Build system

#### Simba build system

The *Simba* build system is based on *GNU Make*.

#### Targets

| Name | Description |
|---|---|
| all | Compile and link the application. |
| clean | Remove all generated files and folders. |
| new | clean + all |
| upload | all + Upload the application to the device. |
| console | Open a serial console on /dev/arduino with baudrate BAUDRATE. |
| run | all + upload + Wait for application output. |
| run-debugger | Run the application in the debugger, break at main. |
| report | Print the test report from a previous run. |
| test | run + report |
| release | Compile with NASSERT=yes. |
| size | Print application size information. |
| help | Show the help. |

#### Variables

There are plenty of make variables used to control the build process. Below is a list of the most frequently used variables. The advanced user may read the make files in make.

| Name | Description |
|---|---|
| SIMBA_ROOT | Path to the *Simba* root folder. |
| BOARD | The `BOARD` variable selects which board to build for. It can be assigned to one of the boards listed *here*. For example, the command to build for *Arduino Due* is `make BOARD=arduino_due`. |
| BAUDRATE | Serial port baudrate used by console and run targets. |
| SERIAL_PORT | Serial port used by console and run targets. |
| VERSION | The application version string. Usually on the form <major>.<minor>.<revision>. |
| SETTINGS_INI | Path to the settings file. |
| INC | Include paths. |
| SRC | Source files (.c, .asm, .rs). |
| CFLAGS_EXTRA | Extra flags passed to the compiler. |
| LDFLAGS_EXTRA | Extra flags passed to the linker. |
| NASSERT | Build the application without assertions. |

### PlatformIO

This section describes Simba specific usage of the PlatformIO build system. Consult the PlatformIO documentation for a general description.

### Application name

To set the application name, create a file called `config.py` in the same folder as your projects `platformio.ini`. The contents of `config.py` can be seen below, and you can change `MyApplicationName` to the name of your application.

```
Import('env')

# Set the Simba application name.
env.Replace(NAME='MyApplicationName')
```

Then add `extra_scripts = config.py` to your `platformio.ini`, as in the example below.

```
[env:my_project]
platform = atmelavr
framework = simba
board = uno
extra_scripts = config.py
```

### Settings

Select your *settings* configuration file with the `platfomio.ini` configuration variable `settings_ini`.

```
[env:my_project]
platform = atmelavr
framework = simba
```

(continues on next page)

(continued from previous page)

```
board = uno
settings_ini = settings.ini
```

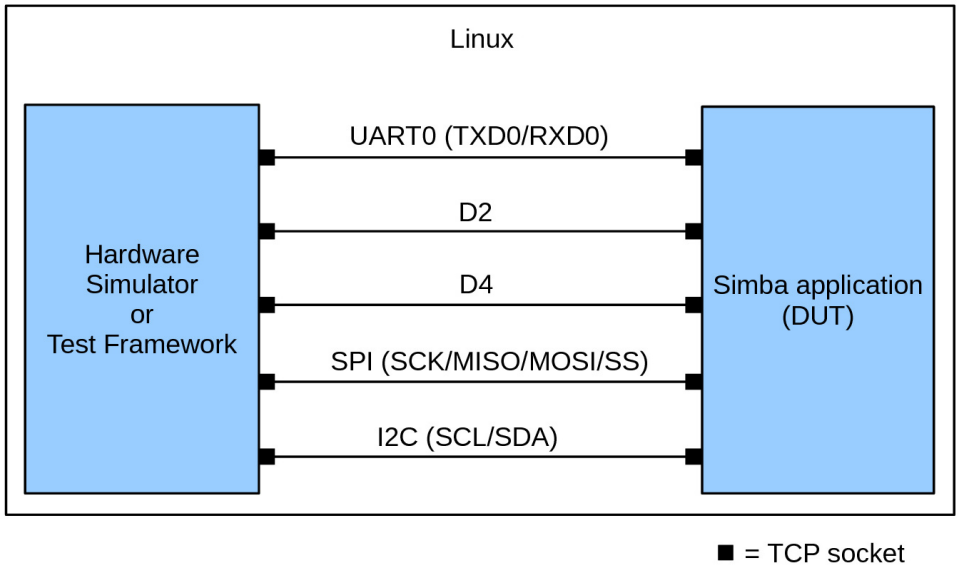
### Socket devices

The Linux socket device drivers implementation allows an external program to simulate the hardware. The external program communicates with the Simba application using TCP sockets, one socket for each device.

The Python script socket_device.py can be used to monitor and send data to a device.

### Arduino Mega example

In this example socket_device.py is the hardware simulator (to the left in the image below), and socket_device is the Simba application (to the right in the image below). The five horizontal lines each represents input and output of one




device. First build and run the linux application with the Arduino Mega pinout...

```
$ make BOARD=linux PINOUT=arduino_mega run
```

...and then, in a second terminal, monitor digital pin 2, `d2`.

```
> socket_device.py pin d2
Connecting to localhost:47000... done.
Requesting pin device d2... done.
$
14:48:10.004512 pin(d2) RX: high
14:48:52.535323 pin(d2) RX: high
14:49:20.123124 pin(d2) RX: low
```

Alternatively, monitor all devices at the same time with the monitor make target.

```
$ make BOARD=linux PINOUT=arduino_mega monitor
socket_device.py monitor
```

(continues on next page)

(continued from previous page)

```
Connecting to localhost:47000... done.
Requesting uart device 0... done.
...
Connecting to localhost:47000... done.
Requesting pin device 2... done.
Connecting to localhost:47000... done.
Requesting pin device 4... done.
...
$
14:51:50.531761 pin(2) RX: low
14:51:50.541784 uart(0) RX: b'\n'
14:51:51.178744 pin(4) RX: high
```

### Python modules

There are two Python modules in the folder bin/socket_device in the Simba repository. Both modules implements the same interface as the default Python module/package with the same name, and can be used to communicate over a socket device instead of using the hardware.

- serial.py implements the pyserial interface.
- can.py implements the python-can interface.

Use the environment variable `PYTHONPATH` to import the socket device modules instead of the default modules/packages.

```
> export PYTHONPATH=$(readlink -f ${SIMBA_ROOT}/bin)
> export PYTHONPATH=${PYTHONPATH}:$(readlink -f ${SIMBA_ROOT}/bin/socket_device)
> bpython3
>>> import serial
>>> serial
<module 'serial' from '/home/erik/workspace/simba/bin/socket_device/serial.py'>
>>> import can
>>> can
<module 'can' from '/home/erik/workspace/simba/bin/socket_device/can.py'>
>>>
```

### Protocol

At startup the Simba application creates a socket and starts listening for clients on TCP port 47000.

### Devices

These drivers supports the socket device protocol at the moment. More to be added when needed.

### Uart

The UART socket is equivalent to a serial port, it streams data to and from the application.

### Pin

Sends `high` or `low` when written to given device. Input is not supported yet.

### Pwm

Sends `frequency=<value>` and `duty_cycle=<value>` when set on given device.

### Can

Sends and receives frames on the format `id=<id>,extended=<extended>,size=<size>,data=<data>`. `<id>` and `<data>` are hexadecimal numbers not prefixed with `0x`. `size` and `<extended>` is a decimal integers.

```
> socket_device.py can 0
Connecting to localhost:47000... done.
Requesting can device 0... done.
$ id=00000005,extended=1,size=2,data=0011<Enter>
14:57:22.344321 can(0) TX: id=00000005,extended=1,size=2,data=0011
14:57:22.346321 can(0) RX: id=00000006,extended=1,size=2,data=0112
```

### I2c

Sends and receives data on the format `address=<address>,size=<size>,data=<data>`. `<address>` is an decimal integer, while `<size>` and `<data>` is a hexadecimal numbers.

```
> socket_device.py i2c 0
Connecting to localhost:47000... done.
Requesting i2c device 0... done.
$
14:57:22.346321 i2c(0) RX: address=0006,size=0003,data=1a2b3c
```

### Device request message

This message is sent to the Simba application to request a device.

```
+---------+---------+----------------+
| 4b type | 4b size | <size>b device |
+---------+---------+----------------+

`device` is the device name as a string without NULL termination.

TYPE  SIZE  DESCRIPTION
--------------------------------------
   1     n  Uart device request.
   3     n  Pin device request.
   5     n  Pwm device request.
   7     n  Can device request.
   9     n  I2c device request.
  11     n  Spi device request.
```

#### Device response message

This message is the response to the request message.

```
+---------+---------+-----------+
| 4b type | 4b size | 4b result |
+---------+---------+-----------+

`result` is zero(0) on success, and otherwise a negative error
code.

Defined error codes are:

   ENODEV(19): No device found matching requested device name.

   EADDRINUSE(98): The requested device is already requested and in
                   use.

TYPE  SIZE  DESCRIPTION
-------------------------------------
   2     4  Uart device response.
   4     4  Pin device response.
   6     4  Pwm device response.
   8     4  Can device response.
  10     4  I2c device response.
  12     4  Spi device response.
```

## 1.3 Developer Guide

This guide is intended for developers of the Simba Embedded Programming Platform. Users are advised to read the *User Guide* instead.

**Contents:**

### 1.3.1 Boards and mcus

A board is the top level configuration entity in the build framework. It contains information about the MCU and the pin mapping.

In turn, the MCU contains information about available devices and clock frequencys in the microcontroller.

See src/boards/ and src/mcus for available configurations.

Only one MCU per board is supported. If there are two MCU:s on one physical board, two board configurations have to be created, one for each MCU.

The porting guide *Porting* shows how to port *Simba* to a new board.

### 1.3.2 Threads and channels

A thread is the basic execution entity. A scheduler controls the execution of threads.

A simple thread that waits for an event from another thread.

```
#include "simba.h"

struct event_t event;

void *my_thread_main(void *arg_p)
{
    uint32_t mask;

    while (1) {
        mask = 0x1;
        event_read(&event, &mask, sizeof(mask));

        std_printf(FSTR("Event received!\r\n"));
    }

    return (NULL);
}
```

Threads usually communicates over channels. There are many kinds of channels; queue, socket and event, to mention a few. All three implementing the same abstract channel interface (see src/sync/chan.h). This abstraction makes channel very powerful as a synchronization primitive. They can be seen as limited functionality file descriptors in linux.

The most common channel is the *queue — Queue channel*. It can be either synchronous or semi-asynchronous. In the synchronous version the writing thread will block until all written data has been read by the reader. In the semi-asynchronous version the writer writes to a buffer within the queue, and only blocks all data does not fit in the buffer. The buffer size is selected by the application.

### 1.3.3 File tree

This is the file tree of the Simba repository.

```
simba                    - this directory
├── 3pp                  - third party products
├── bin                  - executables and scripts
├── doc                  - documentation source
├── environment          - environment setup
├── examples             - example applications
├── LICENSE              - license
├── make                 - build and run files
├── README.rst           - readme
├── setup.sh             - setup script
├── src                  - source code directory
│   ├── alloc            - alloc package
│   ├── boards           - board configurations
│   ├── collections      - collections package
│   ├── debug            - debug package
│   ├── drivers          - drivers package
│   ├── encode           - encode package
│   ├── filesystems      - filesystems package
│   ├── hash             - hash package
│   ├── inet             - inet package
│   ├── kernel           - kernel package
│   ├── mcus             - mcu configurations
│   ├── multimedia       - multimedia package
│   ├── oam              - oam package
│   ├── sync             - sync package
```

(continues on next page)

(continued from previous page)

```
    ├── text              - text package
    ├── simba.h           - includes all package headers
    └── simba.mk          - build system configuration
├── tst                   - test suites
    ├── alloc             - alloc package test suite
    ├── collections       - collections package test suite
    ├── debug             - debug package test suite
    ├── drivers           - drivers package test suite
    ├── encode            - encode package test suite
    ├── filesystems       - filesystems package test suite
    ├── hash              - hash package test suite
    ├── inet              - inet package test suite
    ├── kernel            - kernel package test suite
    ├── multimedia        - multimedia package test suite
    ├── oam               - oam package test suite
    ├── sync              - sync package test suite
    └── text              - text package test suite
└── VERSION.txt           - `Simba` version
```

### 1.3.4 Testing

To ensure high code quility each release is tested extensively by many test suites. The test suites are executed both on native Linux and on many of the supported boards. See *Test suites* for a list of all test suites that are executed before each release.

The native Linux test suites are executed automatically on each commit.

Test result: https://travis-ci.org/eerimoq/simba

Code coverage: https://codecov.io/gh/eerimoq/simba

#### Unit tests

Each module shall have unit tests to verify that the implementation works as expected and that future refactoring does not break legacy.

All unit tests except low level drivers and networking are hardware independent. This makes it possible to use common Linux tools (gcov, valgrind, gdb, etc.) to debug and gather statistics of a module, which is very useful.

For low level drivers where the majority of the code is hardware specific (ports folder), testing on real hardware is important. It's preferable to have a hardware independent test suite with stubbed interfaces for drivers without any port specific code, and having an example application in examples to test on real hardware.

All unit tests are found in the tst folder.

#### Hardware setup

Below is a picture of all supported boards connected to a USB hub. The USB hub is connected to a linux PC (not in the picture) that executes test suites on all boards.

A short description of the setup:

- The DS3231 device (on the breadboard to the left) is connected over i2c to the *Arduino Mega*.
- CAN0 is connected to CAN1 on the *Arduino Due*. The CAN driver is tested by sending frames between the two CAN devices.

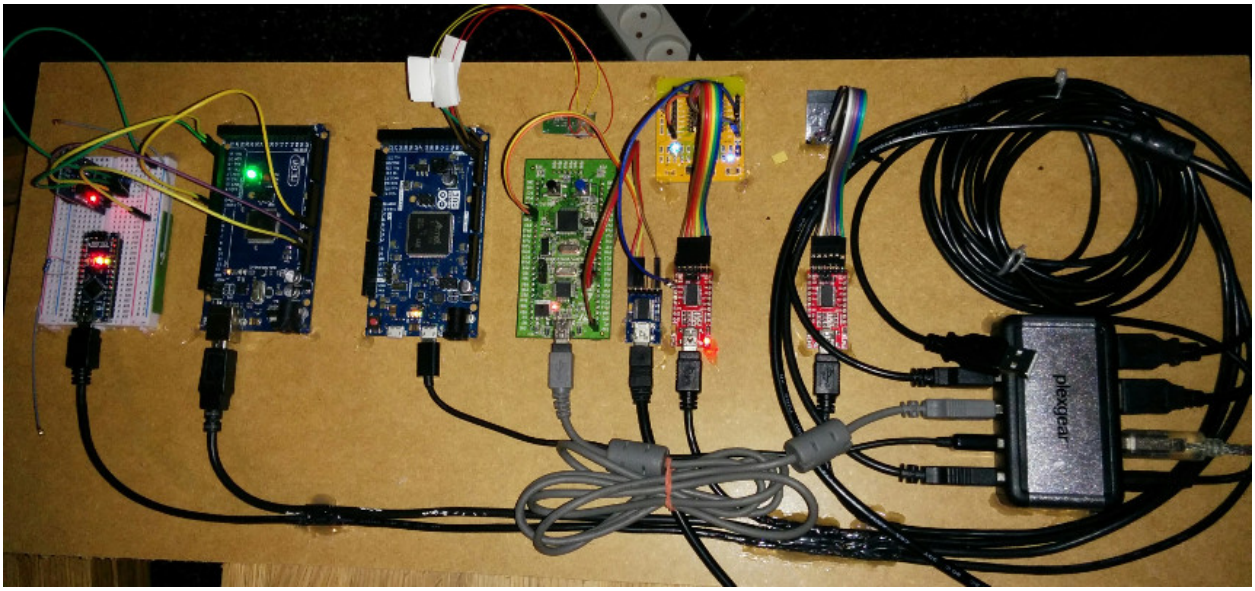



Fig. 1: The boards are (from left to right): *Arduino Nano*, *Arduino Mega*, *Arduino Due*, *STM32VLDISCOVERY*, *ESP-12E Development Board* and *ESP-01*

- The UART of the *STM32VLDISCOVERY* board is connected to a serial to USB adaptor. DTR on the adaptor is used to reset the board.
- The *ESP-12E Development Board* also has a serial to USB adaptor connected. RTS is used to set the board in flashing mode (GPIO0) and DTR is used to reset the board (REST).

### Test suites

Below is a list of all test suites that are executed before every release. They are listed per board.

### Arduino Due

- kernel/sys
- kernel/thrd
- kernel/time
- kernel/timer
- sync/bus
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/fifo

- collections/hash_map
- alloc/circular_heap
- alloc/heap
- text/configfile
- text/std
- text/re
- debug/log
- oam/settings
- oam/shell
- filesystems/fs
- filesystems/spiffs
- encode/base64
- encode/json
- hash/crc
- hash/sha1
- drivers/hardware/basic/chipid
- drivers/hardware/network/can
- drivers/hardware/storage/flash

### Arduino Mega

- kernel/sys
- kernel/thrd
- kernel/time
- kernel/timer
- sync/bus
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/fifo
- collections/hash_map
- alloc/circular_heap
- alloc/heap
- text/configfile

- text/std
- text/re
- debug/log
- oam/settings
- oam/shell
- filesystems/fat16
- filesystems/fs
- encode/base64
- hash/crc
- hash/sha1
- inet/http_websocket_client
- inet/http_websocket_server
- inet/inet
- inet/mqtt_client
- inet/ping
- drivers/hardware/basic/adc
- drivers/hardware/basic/analog_input_pin
- drivers/hardware/various/ds3231
- drivers/hardware/storage/sd
- drivers/hardware/basic/pin

### Arduino Nano

- drivers/hardware/sensors/ds18b20
- drivers/hardware/basic/analog_output_pin
- drivers/hardware/basic/exti
- drivers/hardware/network/owi

### Arduino Pro Micro

- kernel/sys
- kernel/thrd
- kernel/timer

### Arduino Uno

### Arduino Zero

### DEF CON 26 Badge

### ESP-01

### ESP-12E Development Board

- kernel/sys
- kernel/thrd
- kernel/timer

### ESP32-DevKitC

### Adafruit HUZZAH ESP8266 breakout

### Linux

- kernel/sys
- kernel/thrd
- kernel/time
- kernel/timer
- sync/bus
- sync/cond
- sync/chan
- sync/event
- sync/mutex
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/circular_buffer
- collections/fifo
- collections/hash_map
- collections/list
- alloc/circular_heap
- alloc/heap
- text/configfile

- text/emacs
- text/std
- text/re
- debug/log
- debug/harness
- oam/nvm
- oam/service
- oam/settings
- oam/shell
- oam/soam
- oam/upgrade
- oam/upgrade/http
- oam/upgrade/kermit
- oam/upgrade/uds
- filesystems/fat16
- filesystems/fs
- filesystems/spiffs
- encode/base64
- encode/json
- encode/nmea
- hash/crc
- hash/sha1
- inet/http_server
- inet/http_websocket_client
- inet/http_websocket_server
- inet/inet
- inet/isotp
- inet/mqtt_client
- inet/ping
- inet/slip
- inet/ssl
- inet/tftp_server
- multimedia/midi
- drivers/software/network/jtag_soft
- drivers/software/network/xbee
- drivers/software/network/xbee_client

- drivers/software/sensors/dht
- drivers/software/sensors/bmp280
- drivers/software/sensors/hx711
- drivers/software/storage/eeprom_soft
- drivers/software/various/gnss
- science/math
- science/science

### Maple-ESP32

### Nano32

- kernel/sys
- kernel/thrd
- kernel/timer
- sync/bus
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/fifo
- collections/hash_map
- alloc/circular_heap
- text/std
- text/re
- debug/log
- oam/shell
- encode/base64
- encode/json
- hash/crc
- hash/sha1
- inet/http_websocket_client
- inet/http_websocket_server
- inet/inet
- inet/mqtt_client_network
- inet/network_interface/wifi_esp

- inet/ping
- filesystems/fs
- filesystems/spiffs

### NodeMCU

- kernel/sys
- kernel/thrd
- kernel/timer
- sync/bus
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/fifo
- collections/hash_map
- alloc/circular_heap
- text/std
- text/re
- debug/log
- oam/shell
- encode/base64
- encode/json
- hash/crc
- hash/sha1
- inet/http_websocket_client
- inet/http_websocket_server
- inet/inet
- inet/mqtt_client
- inet/network_interface/wifi_esp
- inet/ping
- drivers/hardware/basic/pin
- drivers/hardware/basic/random
- drivers/hardware/basic/power
- filesystems/fs

- filesystems/spiffs

### nRF52840-PDK

### Particle IO Photon

- kernel/sys
- kernel/thrd
- kernel/time
- kernel/timer
- sync/bus
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/fifo
- collections/hash_map
- alloc/circular_heap
- text/std
- text/re
- debug/log
- oam/shell
- encode/base64
- encode/json
- hash/crc
- hash/sha1
- inet/http_websocket_client
- inet/http_websocket_server
- inet/inet
- inet/mqtt_client
- inet/ping

### SPC56D Discovery

- kernel/sys
- kernel/thrd

- kernel/time
- kernel/timer
- sync/bus
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/fifo
- collections/hash_map
- alloc/circular_heap
- text/std
- text/re
- debug/log
- oam/shell
- oam/soam
- encode/base64
- encode/json
- hash/crc
- hash/sha1
- drivers/hardware/storage/eeprom_soft

#### STM32F3DISCOVERY

#### STM32VLDISCOVERY

- kernel/sys
- kernel/thrd
- kernel/timer
- sync/bus
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/fifo

- collections/hash_map
- alloc/circular_heap
- text/std
- text/re
- debug/log
- oam/shell
- encode/base64
- encode/json
- hash/crc
- hash/sha1
- inet/http_websocket_client
- inet/http_websocket_server
- inet/inet
- inet/mqtt_client
- inet/ping
- drivers/hardware/basic/pin
- drivers/hardware/basic/random

### WEMOS D1 mini

### Xvisor Raspberry Pi 3

- kernel/sys
- kernel/thrd
- kernel/time
- kernel/timer
- sync/cond
- sync/event
- sync/mutex
- sync/queue
- alloc/heap
- alloc/circular_heap
- debug/log
- encode/base64
- encode/json
- hash/crc
- hash/sha1
- text/std

- text/emacs

### 1.3.5 Releasing

Follow these steps to create a new release:

1. Write the new version in `VERSION.txt`. The version should hace the format `<major>.<minor>.<revision>`.

   Increment `<major>` for non-backwards compatible changes.

   Increment `<minor>` for new features.

   Increment `<revision>` for bug fixes.

2. Write the new version in `package.json`. This file is used by *PlatformIO 3* to find the current *Simba* release.

3. Run the test suites and generate the documentation and other files.

```
make -s -j8 test-all-boards
make -s -j8 release-test
```

4. Commit the generated files.

5. Generate files for Arduino and PlatformIO releases. The generated archives and Arduino manifests are copied to the release repository.

```
make -s release
```

6. Add, commit and push the Simba Arduino releases in the release repository.

```
(cd ../simba-releases && \
 git add arduino/*/*.zip platformio/*.zip && \
 git commit && \
 git push origin master)
```

7. Start a http server used to download package manifests in the Arduino IDE.

```
(cd make/arduino && python -m SimpleHTTPServer)
```

8. Start the Arduino IDE and add these URL:s in Preferences.

```
http://localhost:8000/avr/package_simba_avr_index.json
http://localhost:8000/esp/package_simba_esp_index.json
http://localhost:8000/esp32/package_simba_esp32_index.json
http://localhost:8000/sam/package_simba_sam_index.json
```

9. Install all four packages and run the blink example for each one of them.

10. Commit the manifests, tag the commit with `<major>.<minor>.<revision>` and push.

```
git commit
git tag <major>.<minor>.<revision>
git push origin master
```

11. Add, commit and push the Simba Arduino package manifests in the release repository.

```
(cd ../simba-releases && \
 git add arduino/*/*.json && \
 git commit && \
 git push origin master)
```

12. Done.

### 1.3.6 Porting

#### Adding a new board

Often the board you want to use in your project is not yet supported by *Simba*. If you are lucky, *Simba* is already ported to the MCU on your board. Just create a folder with you board name in src/boards/ and populate it with the `board.h`, `board.c` and `board.mk`. Also, create board documentation in doc/boards/, and add a pinout image in doc/images/boards/.

The same files as a file tree:

```
simba/
+-- doc/
|   +-- boards/
|   |   +-- <board-name>.rst
|   +-- images/
|       +-- boards/
|           +-- <board-name>-pinout.jpg
+-- src/
    +-- boards/
        +-- <board-name>.h
        +-- <board-name>.c
        +-- <board-name>.mk
```

If *Simba* is not ported to your MCU, the kernel and drivers has to be ported.

#### Kernel

Porting the kernel is a matter of configuring the system tick timer and inplement a few locking primitives. If you are familiar with your CPU, the port can be implemented quickly.

A kernel port is roughly 400 lines of code.

Kernel ports are implemented in src/kernel/ports.

#### Drivers

The required work to port the drivers depends of which drivers you are intrested in. The more drivers you have to port, the longer time it takes, obviously.

A drivers port is roughly 200 lines of code per driver.

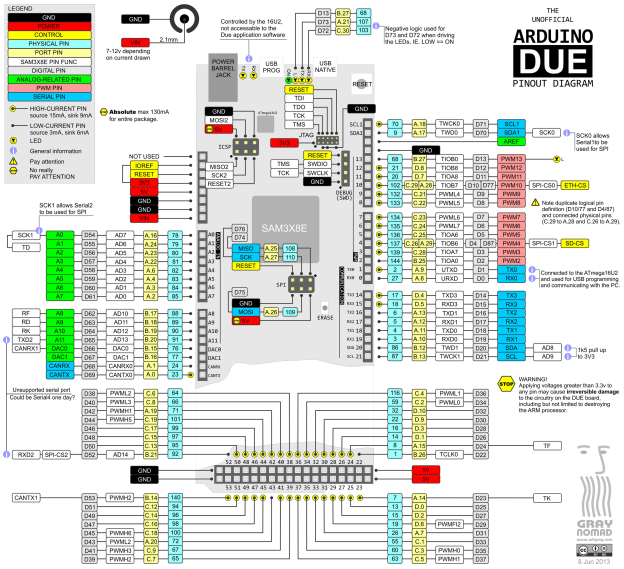
Drivers ports are implemented in src/drivers/ports.

## 1.4 Boards

The boards supported by *Simba*.

### 1.4.1 Arduino Due

#### Pinout




#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

#### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *can — Controller Area Network*
- *chipid — Chip identity*
- *dac — Digital to analog convertion*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *exti — External interrupts*

- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *mcp2515 — CAN BUS chipset*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *random — Random numbers.*
- *sd — Secure Digital memory*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *usb — Universal Serial Bus*
- *usb_host — Universal Serial Bus - Host*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Arduino Due* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 2816 | 1679 |
| default-configuration | 138496 | 10241 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 1 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 1 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 1 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |

Continued on next page

Table 1 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 1 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 1 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 1 |
| CONFIG_MODULE_INIT_DAC | 1 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 1 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |

Continued on next page

Table 1 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 1 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 1 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 1 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |

Continued on next page

Table 1 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SD | 1 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 115200 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x000e0000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 32768 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 1536 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 1536 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |

Continued on next page

Table 1 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 384 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 512 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 1 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 1 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

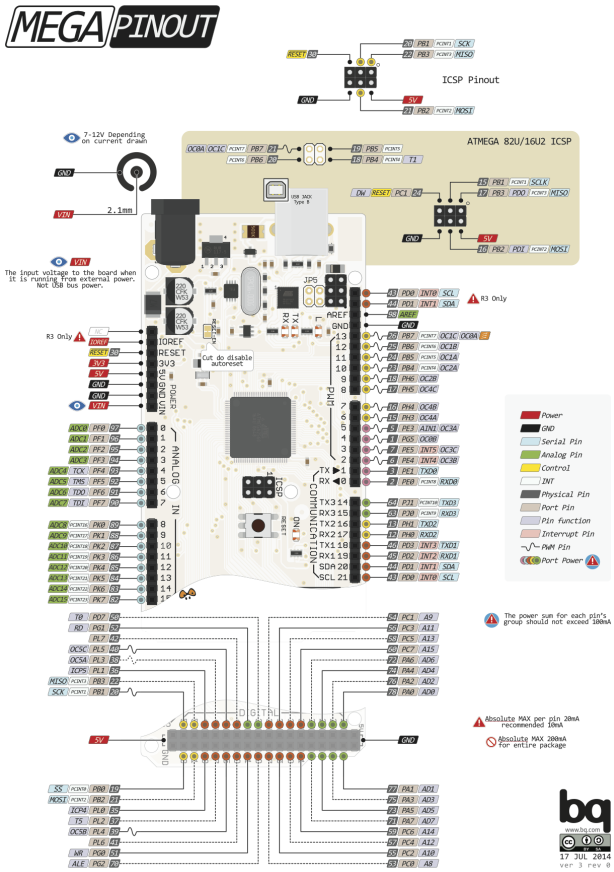
https://www.arduino.cc/en/Main/ArduinoBoardDue

### Mcu

*sam3x8e*

## 1.4.2 Arduino Mega

### Pinout




### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*

- *analog_output_pin — Analog output pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *exti — External interrupts*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *mcp2515 — CAN BUS chipset*
- *nrf24l01 — Wireless communication*
- *owi — One-Wire Interface*
- *pcint — Pin change interrupts*
- *pin — Digital pins*
- *pwm — Pulse width modulation*
- *pwm_soft — Software pulse width modulation*
- *sd — Secure Digital memory*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *watchdog — Hardware watchdog*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Arduino Mega* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.

- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 1730 | 281 |
| default-configuration | 74300 | 3752 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 0 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 0 |

Continued on next page

Table 2 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 1 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 0 |

Continued on next page

Table 2 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 1 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 1 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 1 |
| CONFIG_MODULE_INIT_PWM_SOFT | 1 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 1 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 1 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 1 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 256 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 1 |

Continued on next page

Table 2 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 1 |
| CONFIG_PWM_SOFT | 1 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 1 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |

Continued on next page

Table 2 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 0 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 156 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 256 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 1 |

Continued on next page

Table 2 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

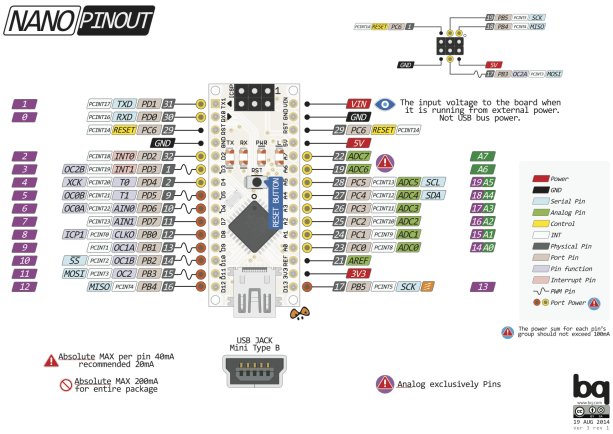
https://www.arduino.cc/en/Main/ArduinoBoardMega

#### Mcu

*atmega2560*

### 1.4.3 Arduino Nano

#### Pinout




#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*

#### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*

- *analog_output_pin — Analog output pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *exti — External interrupts*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *mcp2515 — CAN BUS chipset*
- *nrf24l01 — Wireless communication*
- *owi — One-Wire Interface*
- *pcint — Pin change interrupts*
- *pin — Digital pins*
- *pwm — Pulse width modulation*
- *sd — Secure Digital memory*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *watchdog — Hardware watchdog*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Arduino Nano* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 1540 | 281 |
| default-configuration | 12966 | 603 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 0 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 0 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 0 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |

Continued on next page

Table 3 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FS_FS_COMMAND_APPEND | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 0 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_FORMAT | 0 |
| CONFIG_FS_FS_COMMAND_LIST | 0 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_READ | 0 |
| CONFIG_FS_FS_COMMAND_REMOVE | 0 |
| CONFIG_FS_FS_COMMAND_WRITE | 0 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 1 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 1 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 0 |
| CONFIG_I2C_FS_COMMAND_SCAN | 0 |
| CONFIG_I2C_FS_COMMAND_WRITE | 0 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 0 |
| CONFIG_MCP2515 | 1 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 0 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |

Continued on next page

Table 3 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 1 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 1 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 1 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 1 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 1 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 0 |
| CONFIG_NRF24L01 | 1 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 256 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 0 |
| CONFIG_PCINT | 1 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |

Continued on next page

Table 3 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_PIN_FS_COMMAND_READ | 0 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 0 |
| CONFIG_PIN_FS_COMMAND_WRITE | 0 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 1 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 1 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 0 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 0 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 0 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 0 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 1 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 4 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |

Continued on next page

Table 3 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 0 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 0 |
| CONFIG_SYS_FS_COMMANDS | 0 |
| CONFIG_SYS_LOG_MASK | -1 |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 0 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 0 |
| CONFIG_THRD_FS_COMMANDS | 0 |
| CONFIG_THRD_IDLE_STACK_SIZE | 156 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 256 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 0 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 0 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 0 |
| CONFIG_WATCHDOG | 1 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |

Continued on next page

Table 3 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

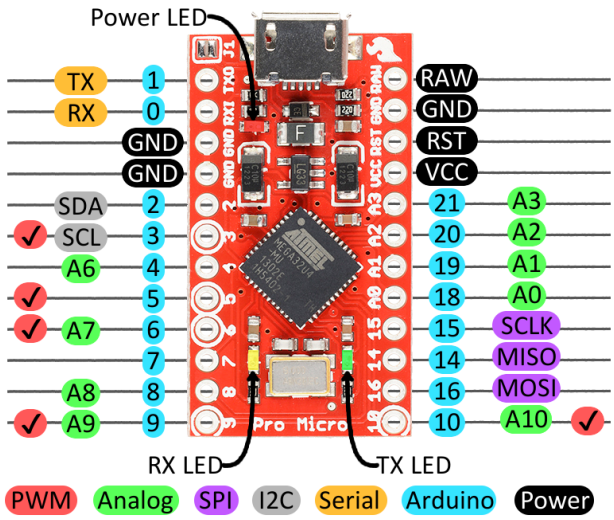
https://www.arduino.cc/en/Main/ArduinoBoardNano

#### Mcu

*atmega328p*

### 1.4.4 Arduino Pro Micro

#### Pinout




#### Enter the bootloader

Recover a bricked board by entering the bootloader.

1. Power up the board.
2. Connect RST to GND for a second to enter the bootloader and stay in it for 8 seconds.

#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *analog_output_pin — Analog output pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *exti — External interrupts*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *mcp2515 — CAN BUS chipset*
- *nrf24l01 — Wireless communication*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *pwm — Pulse width modulation*
- *sd — Secure Digital memory*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *usb — Universal Serial Bus*
- *usb_device — Universal Serial Bus - Device*
- *watchdog — Hardware watchdog*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Arduino Pro Micro* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 6812 | 518 |
| default-configuration | 14510 | 720 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 0 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 0 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |

Continued on next page

Table 4 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 0 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 0 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_FORMAT | 0 |
| CONFIG_FS_FS_COMMAND_LIST | 0 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_READ | 0 |
| CONFIG_FS_FS_COMMAND_REMOVE | 0 |
| CONFIG_FS_FS_COMMAND_WRITE | 0 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 1 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 1 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 0 |
| CONFIG_I2C_FS_COMMAND_SCAN | 0 |
| CONFIG_I2C_FS_COMMAND_WRITE | 0 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 0 |
| CONFIG_MCP2515 | 1 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |

Continued on next page

Table 4 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 0 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 1 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 1 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 1 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 1 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 1 |
| CONFIG_MODULE_INIT_USB_DEVICE | 1 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 1 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 0 |
| CONFIG_NRF24L01 | 1 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |

Continued on next page

Table 4 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 256 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 0 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 0 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 0 |
| CONFIG_PIN_FS_COMMAND_WRITE | 0 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 1 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 1 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 0 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 0 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 0 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 0 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 1 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_USB_CDC |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 4 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |

Continued on next page

Table 4 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 0 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 0 |
| CONFIG_SYS_FS_COMMANDS | 0 |
| CONFIG_SYS_LOG_MASK | -1 |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 0 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 0 |
| CONFIG_THRD_FS_COMMANDS | 0 |
| CONFIG_THRD_IDLE_STACK_SIZE | 156 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 256 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 0 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 1 |
| CONFIG_USB_DEVICE | 1 |

Continued on next page

Table 4 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 0 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 0 |
| CONFIG_WATCHDOG | 1 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

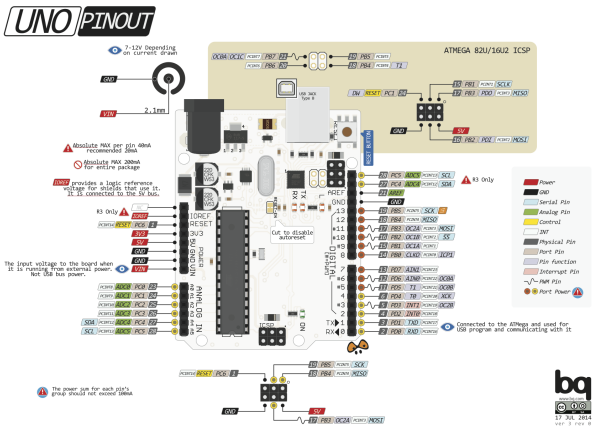
https://www.sparkfun.com/products/12640

#### Mcu

*atmega32u4*

### 1.4.5 Arduino Uno

#### Pinout




#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *analog_output_pin — Analog output pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *exti — External interrupts*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *mcp2515 — CAN BUS chipset*
- *nrf24l01 — Wireless communication*
- *owi — One-Wire Interface*
- *pcint — Pin change interrupts*
- *pin — Digital pins*
- *pwm — Pulse width modulation*
- *sd — Secure Digital memory*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *watchdog — Hardware watchdog*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Arduino Uno* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 1540 | 281 |
| default-configuration | 12966 | 603 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 0 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 0 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |

Continued on next page

Table 5 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 0 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 0 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_FORMAT | 0 |
| CONFIG_FS_FS_COMMAND_LIST | 0 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_READ | 0 |
| CONFIG_FS_FS_COMMAND_REMOVE | 0 |
| CONFIG_FS_FS_COMMAND_WRITE | 0 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 1 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 1 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 0 |
| CONFIG_I2C_FS_COMMAND_SCAN | 0 |
| CONFIG_I2C_FS_COMMAND_WRITE | 0 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 0 |
| CONFIG_MCP2515 | 1 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |

Continued on next page

Table 5 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 0 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 1 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 1 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 1 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 1 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 1 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 0 |
| CONFIG_NRF24L01 | 1 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |

Continued on next page

Table 5 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 256 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 0 |
| CONFIG_PCINT | 1 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 0 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 0 |
| CONFIG_PIN_FS_COMMAND_WRITE | 0 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 1 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 1 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 0 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 0 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 0 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 0 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 1 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 4 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |

Continued on next page

Table 5 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 0 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 0 |
| CONFIG_SYS_FS_COMMANDS | 0 |
| CONFIG_SYS_LOG_MASK | -1 |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 0 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 0 |
| CONFIG_THRD_FS_COMMANDS | 0 |
| CONFIG_THRD_IDLE_STACK_SIZE | 156 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 256 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 0 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |

Continued on next page

Table 5 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 0 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 0 |
| CONFIG_WATCHDOG | 1 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

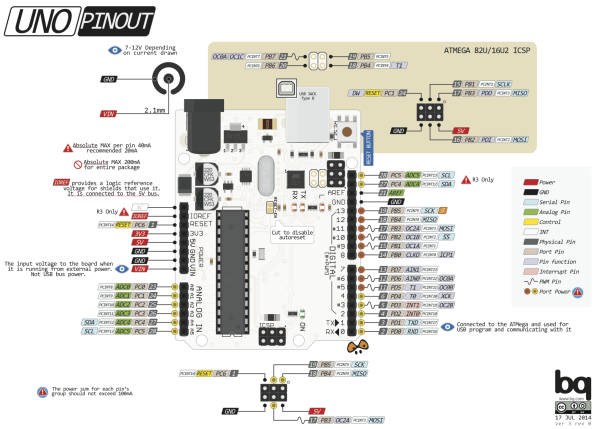
https://www.arduino.cc/en/Main/ArduinoBoardUno

#### Mcu

*atmega328p*

### 1.4.6 Arduino Zero

#### Pinout




**Caution:** This port is under development and does not work in its current state!

#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the Arduino Zero module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 768 | 1092 |
| default-configuration | 768 | 1092 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 0 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 0 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |

Continued on next page

Table 6 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 0 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |

Continued on next page

Table 6 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |

Continued on next page

Table 6 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |

Continued on next page

Table 6 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 384 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 512 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

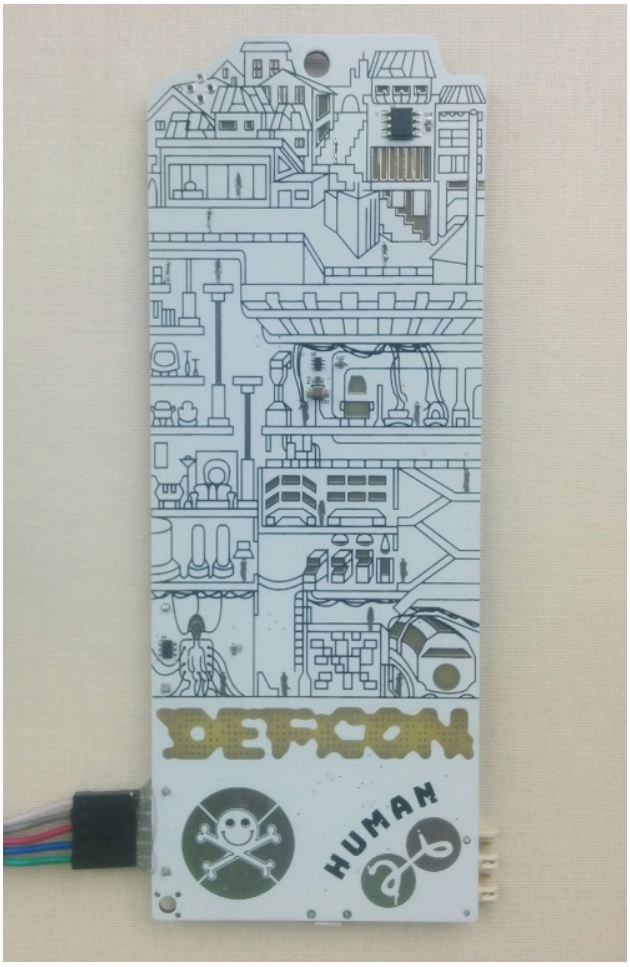
https://www.arduino.cc/en/Main/ArduinoBoardZero

#### Mcu

samd21g18

### 1.4.7 DEF CON 26 Badge

#### Pinout




#### Pin functions

- Serial port in the pairing connector.
- Programming using ICSP pads under battery pack.

```
                    +------------------+
                    |                  |
```

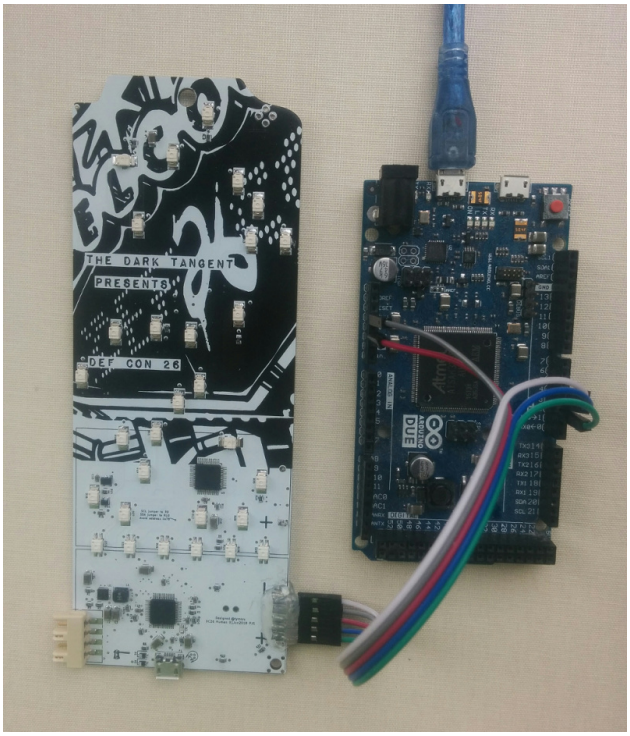
(continues on next page)

(continued from previous page)

```
                  |                    |
                  |                    |
                  |                    |
                  |                    |
                  |                    |
                  | DEF CON 26 Badge   |
                  |                    o--- MCLRN
 PIC-UART-TX ---o                      o--- VDD (3V3)
PIC-UART-GND ---o                      o--- VSS (GND)
                 o                     o--- PGED
 PIC-UART-RX ---o                      o--- PGEC
                  |                    |
                  +--------o---------+
                           |

                         USB
```

### Programming setup




An *Arduino Due* (to the right) acts as programmer, connected to a PC with serial over USB. DEF CON 26 Badge (to the left) with PIC32MM MCU to be programmed.

| Signal | Color | Arduino pin |
|---|---|---|
| MCLRN | white | D4 |
| VDD (3V3) | grey | 3V3 |
| VSS (GND) | purple | GND |
| PGED | blue | D3 |
| PGEC | green | D2 |

Upload ramapp to the Arduino Due, and use pic32tool.py to program the PIC32 MCU.

Inhibit Arduino Due reset when opening the serial port to the programmer on Linux:

```
stty -F /dev/arduino -hup
```

### Component connections

```
D11.1 - U3.23
D11.2 - 3V3

D12.1 - U3.37
D12.2 - 3V3

D16.1 - U3.16
D16.2 - 3V3

U3.18
    U3.15
U3.23

D18.1 - U3.20
D18.2 - 3V3

D20.1 - U3.2
D20.2 - 3V3

D21.1 - U3.7
D21.2 - 3V3

D27.1 - U3.26
D27.2 - 3V3

D28.1 - U3.3
D28.2 - 3V3
D28.3 - U3.5
D28.4 - U3.6

D29.1 - U3.47
D29.2 - 3V3
D29.3 - U3.48
D29.4 - U3.1

D30.1 - U3.35
D30.2 - 3V3
D30.3 - U3.36
D30.4 - U3.37

D31.1 - U3.30
D31.2 - 3V3
D31.3 - U3.31
D31.4 - U3.34

D32.1 - U3.27
D32.2 - 3V3
D32.3 - U3.28
D32.4 - U3.29
```

(continues on next page)

(continued from previous page)

```
D33.1 - U3.23
D33.2 - 3V3
D33.3 - U3.24
D33.4 - U3.25

D35.1 - U3.4
D35.2 - 3V3

D36.1 - U3.32 - D37.1
D36.2 - 3V3

D37.1 - U3.32 - D36.1
D37.2 - 3V3

U3.45 - U2.25 (I2C SDA)
U3.46 - U2.26 (I2C SCL)
```

### U3 - LED driver?

```
        24 23 22 21 20 19 18 17 16 15 14 13
        |  |  |  |  |  |  |  |  |  |  |  |
     +--o--o--o--o--o--o--o--o--o--o--o--o-+
25 --o                                    o-- 12
26 --o                                    o-- 11
27 --o                                    o-- 10
28 --o                                    o-- 9
29 --o                                    o-- 8
30 --o               S2319                o-- 7
31 --o                                    o-- 6
32 --o                                    o-- 5
33 --o                                    o-- 4
34 --o                                    o-- 3
35 --o                                  X o-- 2
36 --o                                    o-- 1
     +--o--o--o--o--o--o--o--o--o--o--o--o-+
        |  |  |  |  |  |  |  |  |  |  |  |
        37 38 39 40 41 42 43 44 45 46 47 48
```

### I2C protocol

### Example transfer

78 01 00 . . .

### Registers?

| Address | Description |
|---|---|
| 0x01 | Dx, off(0) or on(1) |
| 0x02 | Dx, off(0) or on(1) |
| 0x03 | Dx, off(0) or on(1) |

Continued on next page

Table 7 – continued from previous page

| Address | Description |
|---|---|
| 0x04 | Dx, off(0) or on(1) |
| 0x05 | Dx, off(0) or on(1) |
| 0x06 | Dx, off(0) or on(1) |
| 0x07 | Dx, off(0) or on(1) |
| 0x08 | Dx, off(0) or on(1) |
| 0x09 | Dx, off(0) or on(1) |
| 0x0a | Dx, off(0) or on(1) |
| 0x0b | Dx, off(0) or on(1) |
| 0x0c | Dx, off(0) or on(1) |
| 0x0d | Dx, off(0) or on(1) |
| 0x0e | Dx, off(0) or on(1) |
| 0x0f | Dx, off(0) or on(1) |
| 0x10 | Dx, off(0) or on(1) |
| 0x11 | Dx, off(0) or on(1) |
| 0x12 | Dx, off(0) or on(1) |
| 0x13 | Dx, off(0) or on(1) |
| 0x14 | Dx, off(0) or on(1) |
| 0x15 | Dx, off(0) or on(1) |
| 0x16 | Dx, off(0) or on(1) |
| 0x17 | Dx, off(0) or on(1) |
| 0x18 | Dx, off(0) or on(1) |
| 0x19 | Dx, off(0) or on(1) |
| 0x1a | Dx, off(0) or on(1) |
| 0x1b | Dx, off(0) or on(1) |
| 0x1c | Dx, off(0) or on(1) |
| 0x1d | Dx, off(0) or on(1) |
| 0x1e | Dx, off(0) or on(1) |
| 0x1f | Dx, off(0) or on(1) |
| 0x20 | Dx, off(0) or on(1) |
| 0x21 | Dx, off(0) or on(1) |
| 0x22 | Dx, off(0) or on(1) |
| 0x23 | Dx, off(0) or on(1) |
| 0x24 | Dx, off(0) or on(1) |
| 0x26- | Typically 0xff, but unclear what it is. |

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *ds18b20 — One-wire temperature sensor*

- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *icsp_soft — Software ICSP*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *DEF CON 26 Badge* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 4544 | 33227 |
| default-configuration | 129360 | 65997 |

### Default configuration

The communication between the PC and the board is carried over **UART 38400-8-N-1**.

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 0 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DHT_COMMAND_READ | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 0 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |

Continued on next page

Table 8 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_ICSP_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |

Continued on next page

Table 8 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |

Continued on next page

Table 8 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |

Continued on next page

Table 8 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 256 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 2048 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

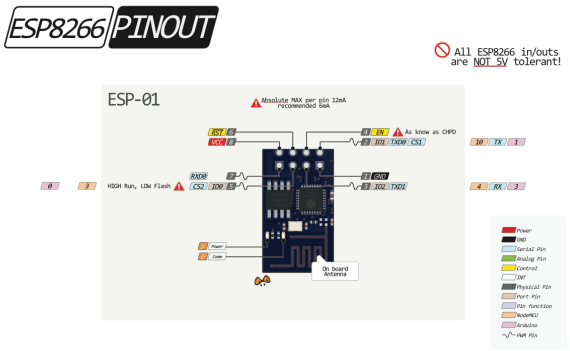
https://www.defcon.org/html/defcon-26/dc-26-index.html

### Mcu

*pic32mm0256gpm048*

## 1.4.8 ESP-01

### Pinout




### Flashing

1. Connect VCC to 3.3 V and GND to ground.
2. Connect GPIO0 to GND.
3. Connect EN/CHPH to 3.3 V.
4. Turn on the power.
5. Upload the software to Flash using esptool.

### Boot from flash

1. Connect VCC to 3.3 V and GND to ground.
2. Connect GPIO0 to 3.3 V.
3. Connect EN/CHPH to 3.3 V.
4. Turn on the power.

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*

- File system.
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *esp_wifi — Espressif WiFi*
- *exti — External interrupts*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *led_7seg_ht16k33 — LED 7-Segment HT16K33*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *power — Power control*
- *pwm_soft — Software pulse width modulation*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *ESP-01* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 277828 | 35716 |
| default-configuration | 325608 | 49556 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |

Continued on next page

Table 9 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 1 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 1 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |

Continued on next page

Table 9 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 1 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 1 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |

Continued on next page

Table 9 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 1 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 1 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 76800 |

Continued on next page

Table 9 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x0006b000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 0x10000 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 1536 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 1536 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 1 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 768 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |

Continued on next page

Table 9 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

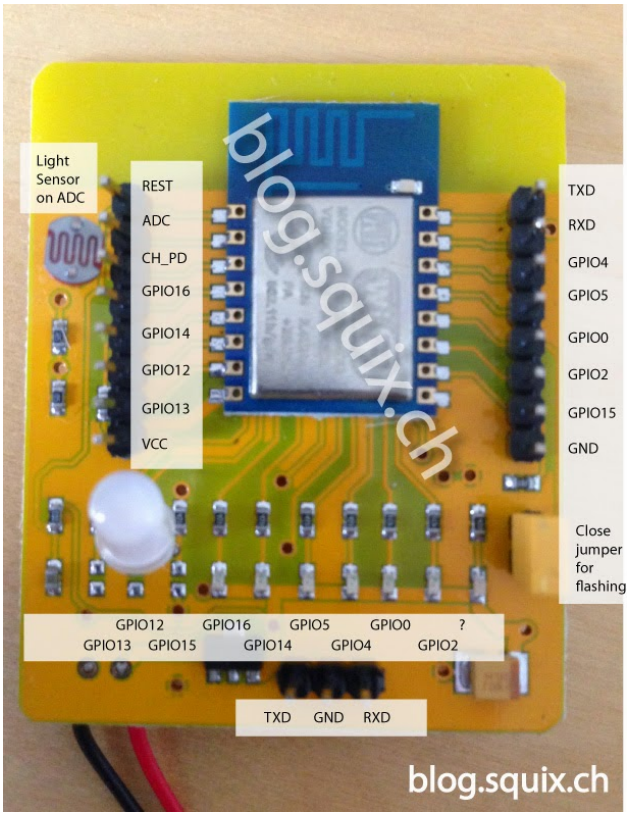
http://espressif.com

### Mcu

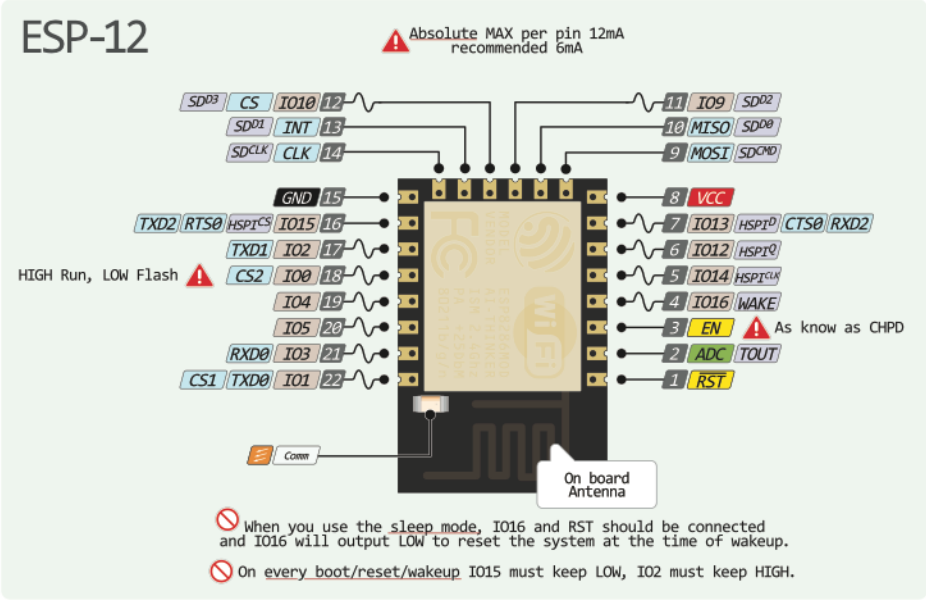
*esp8266*

### 1.4.9 ESP-12E Development Board

#### Pinout




#### ESP-12 pinout



### Flashing

1. Connect 3.3 V to VCC and ground to GND.
2. Attach the flash jumper (to the right in the picture).
3. Turn on the power.
4. Upload the software to Flash using esptool.
5. The application starts automatically when the download is completed.

### Hardware

- 3.3 V power supply and logical level voltage.
- Boot message at 76800 buad on a virgin board. Blue, red and RGB LEDs turned on.
- 4 MB Flash.

How to determine the Flash size:

```
$ python esptool.py --port /dev/ttyUSB0 flash_id
Connecting...
head: 0 ;total: 0
erase size : 0
Manufacturer: e0
Device: 4016
```

Device 4016 gives a Flash of size `2 ^ (16 - 1) / 8 = 4096 kB = 4 MB`.

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*

- *esp_wifi — Espressif WiFi*
- *exti — External interrupts*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *led_7seg_ht16k33 — LED 7-Segment HT16K33*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *power — Power control*
- *pwm_soft — Software pulse width modulation*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *ESP-12E Development Board* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 277828 | 35716 |
| default-configuration | 325724 | 49592 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 1 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |

Continued on next page

Table 10 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 1 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |

Continued on next page

Table 10 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 1 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 1 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 1 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |

Continued on next page

Table 10 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 1 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 76800 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x00300000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 0xFB000 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 1536 |

Continued on next page

Table 10 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 1536 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 1 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 768 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

http://espressif.com

#### Mcu

*esp8266*

### 1.4.10 ESP32-DevKitC

#### Pinout

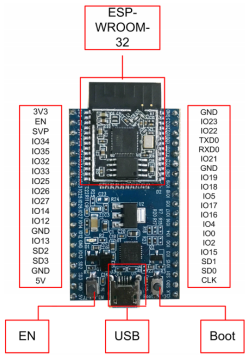



Figure 1-1. ESP32-DevKitC Layout

#### Default pin mapping

Here is a list of additional pin mappings not part of the picture above.

| Device function | GPIO |
|---|---|
| LED | 16 |
| UART 0 TX | 1 |
| UART 0 RX | 3 |
| UART 1 TX | 10 |
| UART 1 RX | 9 |
| UART 2 TX | 17 |
| UART 2 RX | 16 |
| CAN TX | 16 |
| CAN RX | 17 |

#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *can — Controller Area Network*
- *dac — Digital to analog convertion*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *esp_wifi — Espressif WiFi*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *ws2812 — NeoPixels*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *ESP32-DevKitC* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.

- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 91400 | 9680 |
| default-configuration | 361484 | 84420 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 1 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 1 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |

Continued on next page

Table 11 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 1 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 1 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 1 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |

Continued on next page

Table 11 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 1 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 1 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |

Continued on next page

Table 11 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 115200 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 512 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x00300000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 0x20000 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |

Continued on next page

Table 11 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 4096 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 4096 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 8192 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 2048 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |

Continued on next page

Table 11 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 1 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

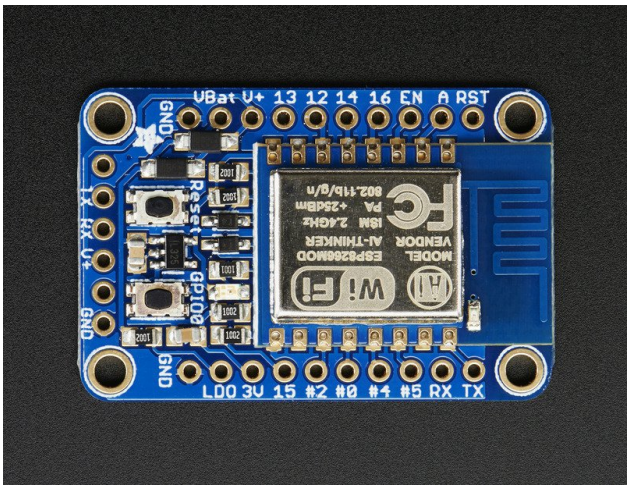
https://espressif.com/en/products/hardware/esp32-devkitc/overview

#### Mcu

*esp32*

### 1.4.11 Adafruit HUZZAH ESP8266 breakout

#### Pinout




#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *esp_wifi — Espressif WiFi*
- *exti — External interrupts*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *led_7seg_ht16k33 — LED 7-Segment HT16K33*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *power — Power control*
- *pwm_soft — Software pulse width modulation*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Adafruit HUZZAH ESP8266 breakout* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 277828 | 35716 |
| default-configuration | 324904 | 49032 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |

Continued on next page

Table 12 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 1 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |

Continued on next page

Table 12 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 1 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 1 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |

Continued on next page

Table 12 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 1 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 1 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 76800 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |

Continued on next page

Table 12 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 1536 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 1536 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 1 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 768 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |

Continued on next page

Table 12 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

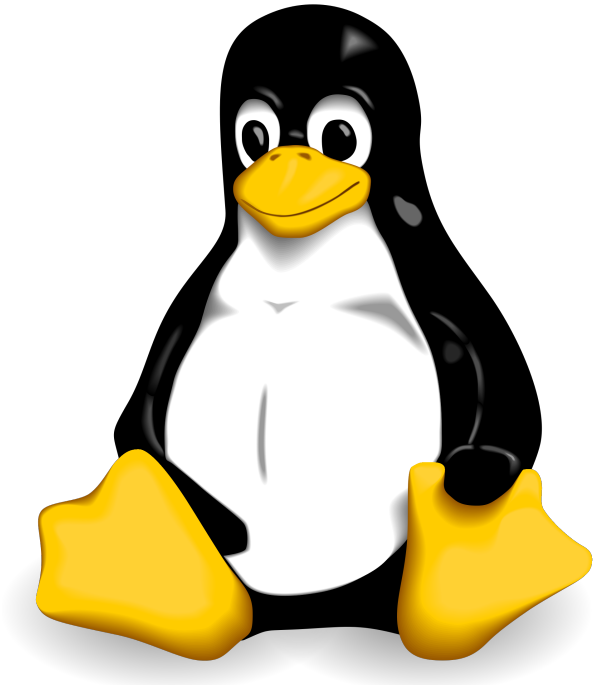
https://www.adafruit.com/product/2471

#### Mcu

*esp8266*

### 1.4.12 Linux

#### Pinout



### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *analog_output_pin — Analog output pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *can — Controller Area Network*
- *dac — Digital to analog convertion*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *exti — External interrupts*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *pwm — Pulse width modulation*
- *pwm_soft — Software pulse width modulation*
- *random — Random numbers.*
- *sd — Secure Digital memory*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*

- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Linux* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 156712 | 351120 |
| default-configuration | 412207 | 521216 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 1 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 1 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |

Continued on next page

Table 13 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 32 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 1024 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 32 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 1 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |

Continued on next page

Table 13 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 1 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 1 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 1 |
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 1 |
| CONFIG_MODULE_INIT_PWM_SOFT | 1 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 1 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 1 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 1 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |

Continued on next page

Table 13 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 1 |
| CONFIG_PWM_SOFT | 1 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 1 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 1028 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |

Continued on next page

Table 13 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 512 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 1 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 0 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 2048 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 2048 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |

Continued on next page

Table 13 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

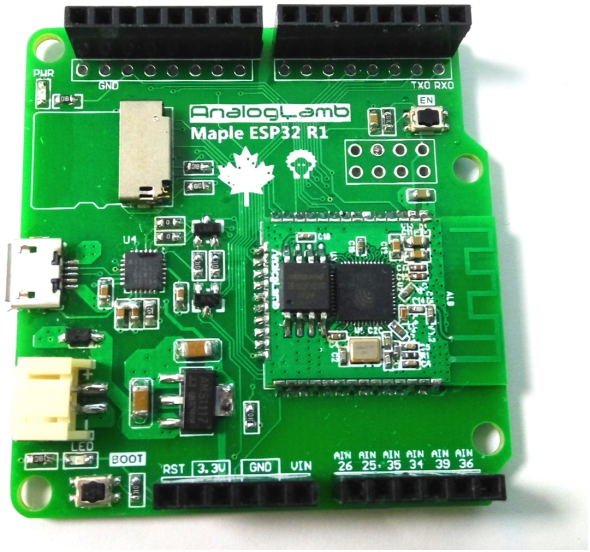
http://www.kernel.org

### Mcu

*linux*

### 1.4.13 Maple-ESP32

#### Pinout




#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

#### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *can — Controller Area Network*
- *dac — Digital to analog convertion*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *esp_wifi — Espressif WiFi*

- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *ws2812 — NeoPixels*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Maple-ESP32* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 91400 | 9680 |
| default-configuration | 361480 | 84420 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |

Continued on next page

Table 14 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 1 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 1 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |

Continued on next page

Table 14 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 1 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 1 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 1 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |

Continued on next page

Table 14 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 1 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 1 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |

Continued on next page

Table 14 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 115200 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 512 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x00300000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 0x20000 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 4096 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 4096 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |

Continued on next page

Table 14 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 8192 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 2048 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 1 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

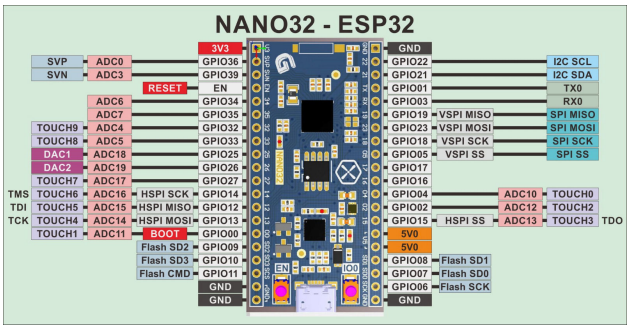
http://www.analoglamb.com/product/maple-esp32/

### Mcu

*esp32*

## 1.4.14 Nano32

### Pinout




### Default pin mapping

Here is a list of additional pin mappings not part of the picture above.

| Device function | GPIO |
|---|---|
| UART 0 TX | 1 |
| UART 0 RX | 3 |
| UART 1 TX | 10 |
| UART 1 RX | 9 |
| UART 2 TX | 17 |
| UART 2 RX | 16 |
| CAN TX | 16 |
| CAN RX | 17 |

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*

- *can — Controller Area Network*
- *dac — Digital to analog convertion*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *esp_wifi — Espressif WiFi*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *ws2812 — NeoPixels*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Nano32* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 91400 | 9680 |
| default-configuration | 361476 | 84420 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 1 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 1 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |

Continued on next page

Table 15 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 1 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 1 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 1 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |

Continued on next page

Table 15 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 1 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 1 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |

Continued on next page

Table 15 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 115200 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 512 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x00300000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 0x20000 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 4096 |

Continued on next page

Table 15 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 4096 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 8192 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 2048 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 1 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

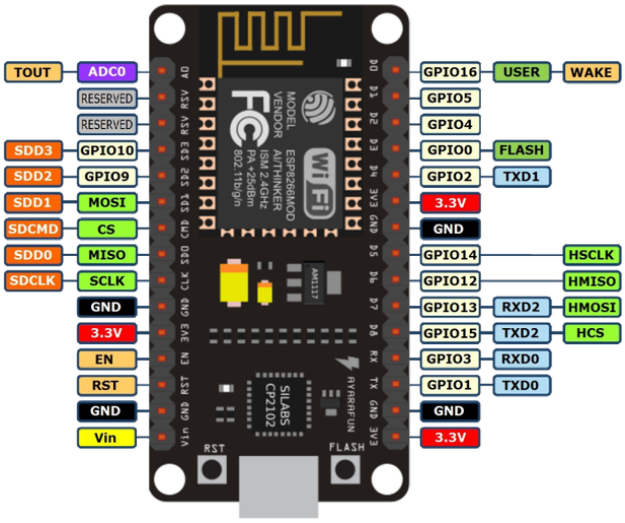
http://esp32.de

### Mcu

*esp32*

## 1.4.15 NodeMCU

### Pinout




### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*

- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *esp_wifi — Espressif WiFi*
- *exti — External interrupts*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *led_7seg_ht16k33 — LED 7-Segment HT16K33*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *power — Power control*
- *pwm_soft — Software pulse width modulation*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *NodeMCU* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 277828 | 35716 |
| default-configuration | 325760 | 49560 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 1 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |

Continued on next page

Table 16 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 1 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |

Continued on next page

Table 16 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 1 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 1 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 1 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |

Continued on next page

Table 16 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 1 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 76800 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x00300000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 0xFB000 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 1536 |

Continued on next page

Table 16 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 1536 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 1 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 768 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

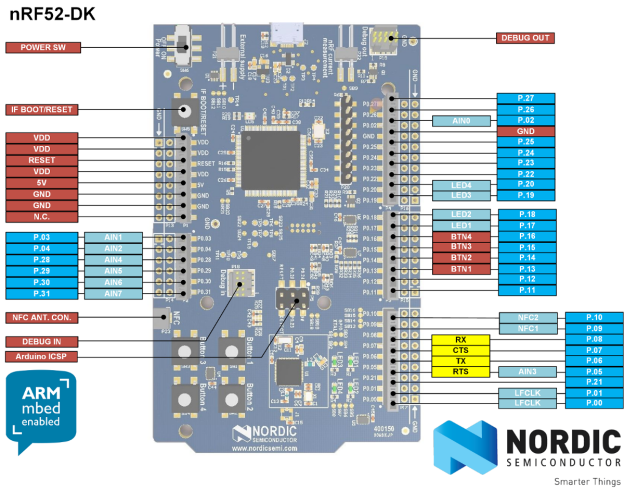
http://www.nodemcu.com

### Mcu

*esp8266*

## 1.4.16 nRF52840-PDK

### Pinout




**Caution:** This port is under development and does not work in its current state!

### Setup

Download and install the *nRF5x Command Line Tools* from https://www.nordicsemi.com.

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *ds18b20 — One-wire temperature sensor*

- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *nRF52840-PDK* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 2048 | 1679 |
| default-configuration | 72448 | 5761 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 0 |

Continued on next page

Table 17 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |

Continued on next page

Table 17 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |

Continued on next page

Table 17 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |

Continued on next page

Table 17 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |

Continued on next page

Table 17 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 384 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 512 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

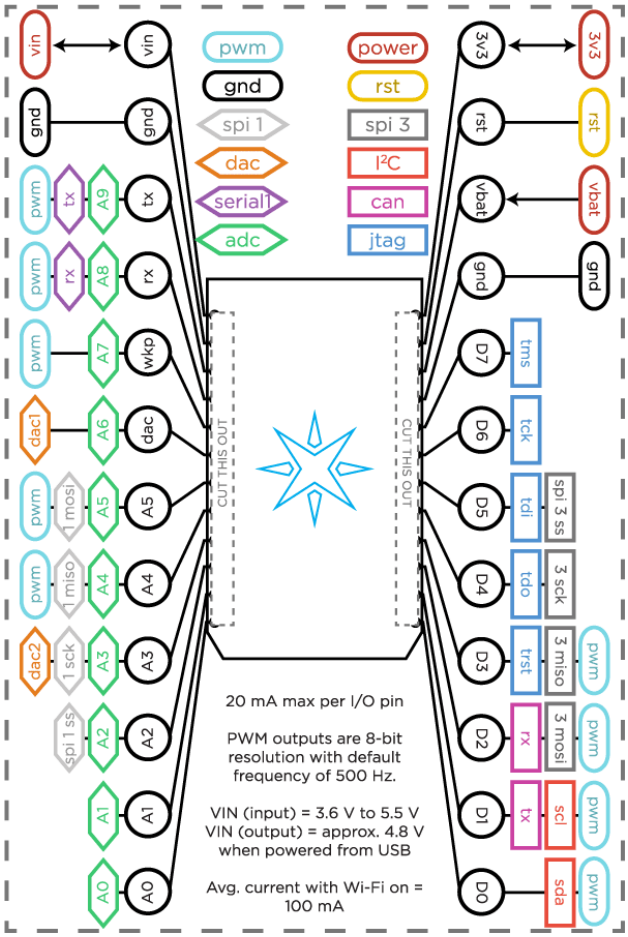
http://www.mouser.se/new/nordicsemiconductor/nordic-nRF52840-dev-kit/

### Mcu

*nrf52840*

## 1.4.17 Particle IO Photon

### Pinout



### Detailed pinout

### Right side pins

| USB | Pin | Exposed Functions | | | | | STM32 Pin | PØ Pin # | PØ Pin Name |
|---|---|---|---|---|---|---|---|---|---|
| PHOTON | 3V3 | 3V3 | | | | | | | |
| | RST | RST | | | | | E8 | 26 | MICRO_RST_N |
| | VBAT | VBAT | | | | | A9 | 28 | VBAT |
| | GND | GND | | | | | | | |
| | D7 | JTAG_TMS | | | | | PA13 | 44 | MICRO_JTAG_TMS |
| | D6 | JTAG_TCK | | | | | PA14 | 40 | MICRO_JTAG_TCK |
| | D5 | JTAG_TDI | SPI3_SS | | | I2S3_WS | PA15 | 43 | MICRO_JTAG_TDI |
| | D4 | JTAG_TDO | SPI3_SCK | | | I2S3_SCK | PB3 | 41 | MICRO_JTAG_TDO |
| | D3 | JTAG_TRST | SPI3_MISO | | TIM3_CH1 | | PB4 | 42 | MICRO_JTAG_TRSTN |
| | D2 | | SPI3_MOSI | CAN2_RX | TIM3_CH2 | I2S3_SD | PB5 | 3 | MICRO_GPIO_5 |
| | D1 | SCL | | CAN2_TX | TIM4_CH1 | | PB6 | 5 | MICRO_GPIO_3 |
| | D0 | SDA | | | TIM4_CH2 | | PB7 | 4 | MICRO_GPIO_4 |

### Left side pins

| Pin | USB | Exposed Functions | | | | | STM32 Pin | PØ Pin # | PØ Pin Name |
|---|---|---|---|---|---|---|---|---|---|
| VIN | PHOTON | VIN | | | | | | | |
| GND | | GND | | | | | | | |
| TX | | | | USART1_TX | TIM1_CH2 | | PA9 | 39 | MICRO_UART_TX |
| RX | | | | USART1_RX | TIM1_CH3 | | PA10 | 38 | MICRO_UART_RX |
| WKP | | ADC0 | | | TIM5_CH1 | | PA0 | 27 | MICRO_WKUP |
| DAC | | ADC4 | | | | DAC1 | PA4 | 22 | MICRO_SPI_SSN |
| A5 | | ADC7 | SPI1_MOSI | | TIM3_CH2 | | PA7 | 23 | MICRO_SPI_MOSI |
| A4 | | ADC6 | SPI1_MISO | | TIM3_CH1 | | PA6 | 25 | MICRO_SPI_MISO |
| A3 | | ADC5 | SPI1_SCK | | | DAC2 | PA5 | 24 | MICRO_SPI_SCK |
| A2 | | ADC12 | SPI1_SS | | | | PC2 | 2 | MICRO_GPIO_6 |
| A1 | | ADC13 | | | | | PC3 | 1 | MICRO_GPIO_7 |
| A0 | | ADC15 | | | | | PC5 | 54 | MICRO_GPIO_8 |

### User I/O

| | User I/O | Photon Pin # | | Exposed Functions | | STM32 Pin | PØ Pin # | PØ Pin Name |
|---|---|---|---|---|---|---|---|---|
| PHOTON | RGB LED - RED | 27 | | TIM2_CH2 | | PA1 | 8 | MICRO_GPIO_0 |
| | RGB LED - GREEN | 28 | | TIM2_CH3 | | PA2 | 7 | MICRO_GPIO_1 |
| | RGB LED - BLUE | 29 | | TIM2_CH4 | | PA3 | 6 | MICRO_GPIO_2 |
| | Setup Button | 26 | | TIM3_CH2 | I2S3_MCK | PC7 | 53 | MICRO_GPIO_9 |
| | Reset Button | 23 | | | | E8 | 26 | MICRO_RST_N |
| | USB Data+ | 31 | | | | PB15 | 51 | MICRO_USB_HS_DP |
| | USB Data- | 30 | | | | PB14 | 52 | MICRO_USB_HS_DM |
| | SMPS Enable | 25 | | | | | | |
| | Peripheral Key | ADC | SPI | PWM/Servo/Tone | | | | |
| | | JTAG | SPI1 | I2S | DAC | | | |
| | | I2C/Wire | Serial1 | CAN | | | | |

### Prerequisities

Install the dfu-utility.

```
git clone git://git.code.sf.net/p/dfu-util/dfu-util
cd dfu-util
sudo apt-get build-dep dfu-util
./autogen.sh
./configure
make
sudo make install
cd ..

# Give users access to the device.
sudo cp simba/environment/udev/49-photon.rules /etc/udec/rules.d
```

### Flashing

The Photon must enter DFU mode before software can be uploaded to it. It's recommended to use the manual method to verify that software can be successfully uploaded to the board, and then start using the automatic method to reduce the manual work for each software upload.

### Automatic (recommended)

- Connect DTR on the serial adapter to the RST pin on the Photon.
- Connect RTS on the serial adapter to the SETUP pad on the bottom side of the Photon. This requires soldering a cable to the SETUP pad.

Upload the software with `make BOARD=photon upload`.

### Manual

To enter DFU Mode:

1. Hold down the RESET and SETUP buttons.
2. Release only the RESET button, while holding down the SETUP button.
3. Wait for the LED to start flashing yellow (it will flash magenta first).
4. Release the SETUP button.

NOTE: Do **not** connect DTR and/or RTS using manual upload. They must only be connected using the automatic method.

Upload the software with `make BOARD=photon upload`.

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Particle IO Photon* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 2668 | 1679 |
| default-configuration | 73576 | 5937 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 0 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |

Continued on next page

Table 18 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |

Continued on next page

Table 18 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |

Continued on next page

Table 18 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |

Continued on next page

Table 18 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 384 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 512 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |

Continued on next page

Table 18 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

#### Homepage

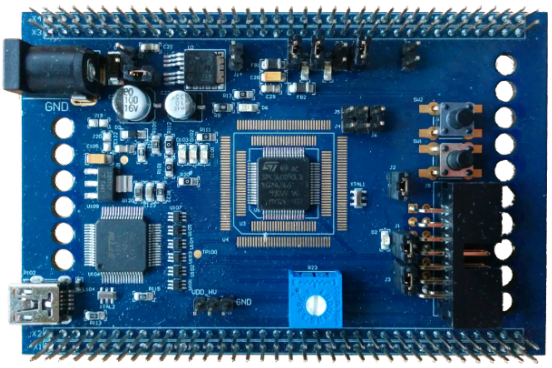
https://docs.particle.io/datasheets/photon-datasheet/

#### Mcu

*stm32f205rg*

### 1.4.18 SPC56D Discovery

#### Pinout



### Pin functions

These are the default pin functions in Simba.

| List | Index | Pin | Function |
|---|---|---|---|
| X1 | 12 | PB0 | CAN TX |
| X2 | 9 | PB1 | CAN RX |
| X1 | 10 | PB2 | UART0 TX |
| X2 | 10 | PB3 | UART0 RX |
| X4 | 6 | PC2 | LED (D8) |

### Toolchain

Download S32 Design Studio for Power v1.1 for Linux and install it.

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *can — Controller Area Network*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*

- *watchdog — Hardware watchdog*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *SPC56D Discovery* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 13884 | 947 |
| default-configuration | 87004 | 5821 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 0 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 1 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |

Continued on next page

Table 19 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 0 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_FORMAT | 0 |
| CONFIG_FS_FS_COMMAND_LIST | 0 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 0 |
| CONFIG_FS_FS_COMMAND_READ | 0 |
| CONFIG_FS_FS_COMMAND_REMOVE | 0 |
| CONFIG_FS_FS_COMMAND_WRITE | 0 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |

Continued on next page

Table 19 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 1 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |

Continued on next page

Table 19 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_WATCHDOG | 1 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 1 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 1028 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |

Continued on next page

Table 19 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 115200 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 1536 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 256 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 156 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |

Continued on next page

Table 19 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 1 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

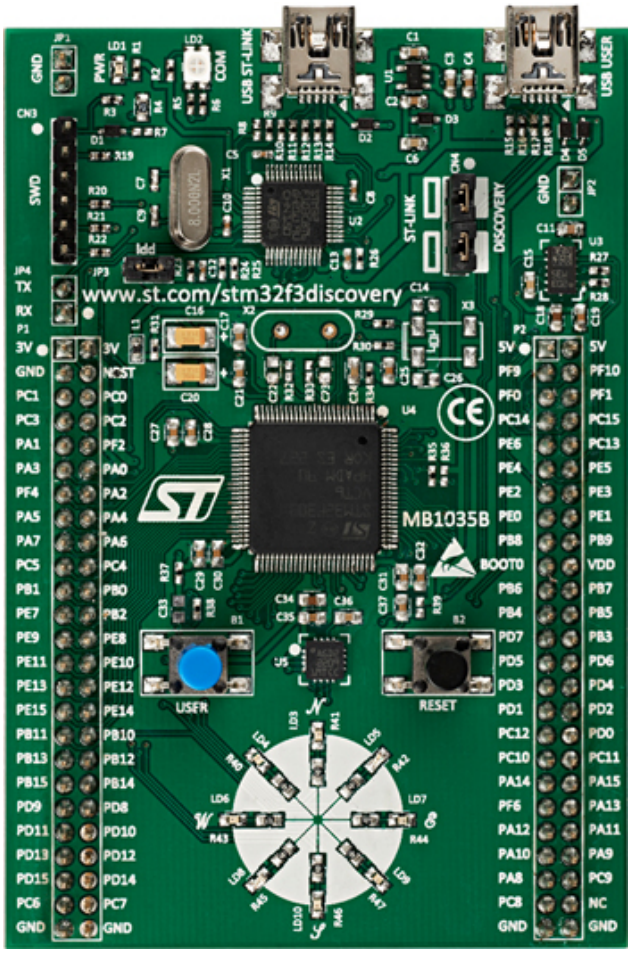
http://www.st.com/en/evaluation-tools/spc56d-discovery.html

### Mcu

*spc56d40l1*

### 1.4.19 STM32F3DISCOVERY

#### Pinout




#### Pin functions

These are the default pin functions in Simba.

| Function | Pin |
|---|---|
| UART0 TX | PA9 |
| UART0 RX | PA10 |
| UART1 TX | PA2 |
| UART1 RX | PA3 |
| UART2 TX | PB10 |
| UART2 RX | PB11 |
| SPI0 SCK | PA5 |
| SPI0 MISO | PA6 |
| SPI0 MOSI | PA7 |
| SPI1 SCK | PA13 |
| SPI1 MISO | PA14 |
| SPI1 MOSI | PA15 |
| SPI2 SCK | PC10 |
| SPI2 MISO | PC11 |
| SPI2 MOSI | PC12 |
| I2C0 SCL | PB8 |
| I2C0 SDA | PB9 |
| I2C1 SCL | PF0 |
| I2C1 SDA | PF1 |
| CAN TX | PD1 |
| CAN RX | PD0 |

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*

- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *STM32F3DISCOVERY* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 2816 | 1679 |
| default-configuration | 72704 | 5441 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 0 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |

Continued on next page

Table 20 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |

Continued on next page

Table 20 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |

Continued on next page

Table 20 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | “$ “ |

Continued on next page

Table 20 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |

Continued on next page

Table 20 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 384 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 512 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

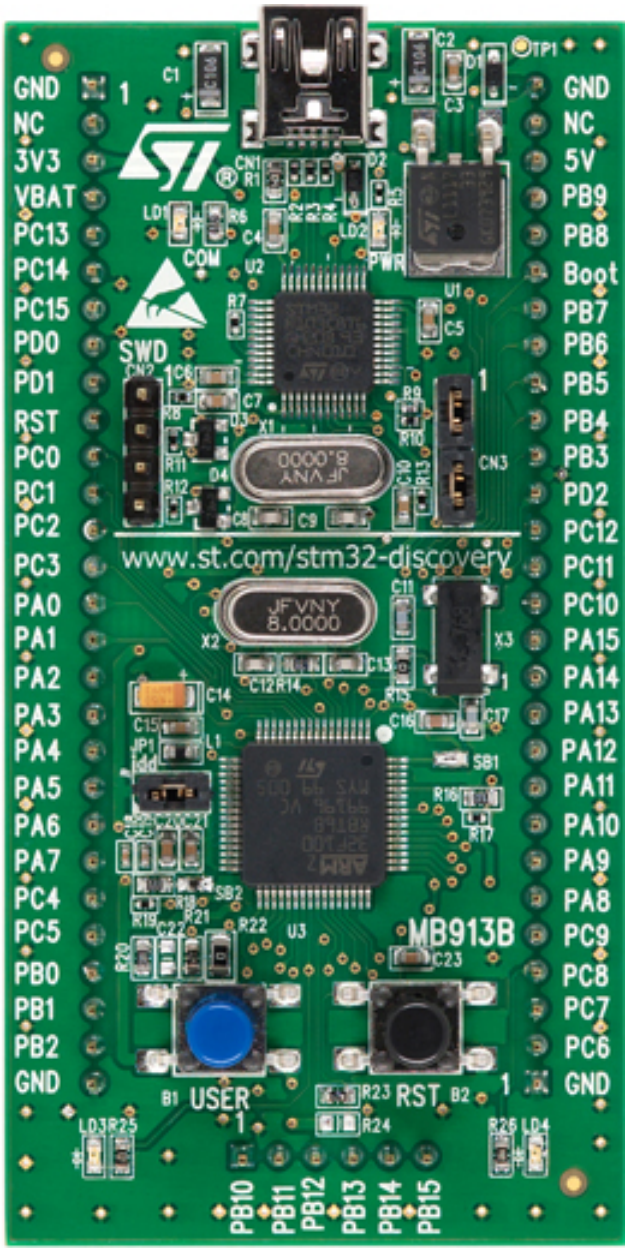
http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-kits/stm32f3discovery.html

### Mcu

*stm32f303vc*

### 1.4.20 STM32VLDISCOVERY

#### Pinout




#### st-link

```
sudo apt install libusb-1.0-0-dev
git clone https://github.com/eerimoq/stlink
./autogen.sh
./configure
make
sudo cp etc/udev/rules.d/49* /etc/udev/rules.d
```

(continues on next page)

(continued from previous page)

```
udevadm control --reload-rules
udevadm trigger

modprobe -r usb-storage && modprobe usb-storage quirks=483:3744:i

st-util -1
arm-none-eabi-gdb app.out
$ target extended-remote localhost:4242
```

Plug in the board in the PC.

### Pin functions

These are the default pin functions in Simba.

| Function | Pin |
|---|---|
| UART0 TX | PA9 |
| UART0 RX | PA10 |
| UART1 TX | PA2 |
| UART1 RX | PA3 |
| UART2 TX | PC10 |
| UART2 RX | PC11 |
| SPI0 SCK | PA5 |
| SPI0 MISO | PA6 |
| SPI0 MOSI | PA7 |
| I2C0 SCL | PB8 |
| I2C0 SDA | PB9 |

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*

- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *STM32VLDISCOVERY* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 2816 | 1679 |
| default-configuration | 73984 | 5937 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 0 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |

Continued on next page

Table 21 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |

Continued on next page

Table 21 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |

Continued on next page

Table 21 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |

Continued on next page

Table 21 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | "$ " |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 768 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 840 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |

Continued on next page

Table 21 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 384 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 512 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

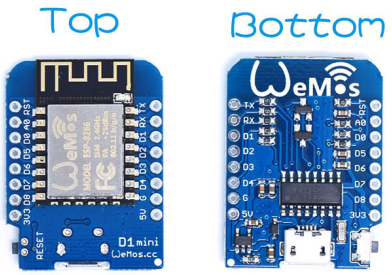
http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-kits/stm32vldiscovery.html?sc=internet/evalboard/product/250863.jsp

### Mcu

*stm32f100rb*

### 1.4.21 WEMOS D1 mini

#### Pinout




#### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- File system.
- *Debug shell.*

#### Drivers

Supported drivers for this board.

- *adc — Analog to digital convertion*
- *analog_input_pin — Analog input pin*
- *bmp280 — BMP280 temperature and pressure sensor*
- *dht — DHT temperature and humidity sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *esp_wifi — Espressif WiFi*
- *exti — External interrupts*
- *flash — Flash memory*
- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*

- *jtag_soft — Software JTAG*
- *led_7seg_ht16k33 — LED 7-Segment HT16K33*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *power — Power control*
- *pwm_soft — Software pulse width modulation*
- *random — Random numbers.*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *uart_soft — Software Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *WEMOS D1 mini* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 277828 | 35716 |
| default-configuration | 324940 | 49048 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 1 |
| CONFIG_ALIGNMENT | 0 |
| CONFIG_ANALOG_INPUT_PIN | 1 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 0 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |

Continued on next page

Table 22 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 1 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 1 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 1 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 1 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |

Continued on next page

Table 22 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 64 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 1 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 1 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 1 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 1 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 1 |
| CONFIG_MODULE_INIT_EXTI | 1 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 1 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 1 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 1 |
| CONFIG_MODULE_INIT_POWER | 1 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 1 |
| CONFIG_MODULE_INIT_RANDOM | 1 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |

Continued on next page

Table 22 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 1 |
| CONFIG_MODULE_INIT_SPI | 1 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 1 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 0 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 1 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 0 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 1 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 1 |
| CONFIG_RANDOM | 1 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |

Continued on next page

Table 22 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | "$ " |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 1 |
| CONFIG_SPIFFS | 1 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 76800 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 1 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0x00300000 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 0xFB000 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 1 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 1536 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 1536 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 1 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |

Continued on next page

Table 22 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 768 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 768 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

https://wiki.wemos.cc/products:d1:d1_mini

### Mcu

*esp8266*

## 1.4.22 Xvisor Raspberry Pi 3

### Pinout

| Peripherals | GPIO | Particle | Pin # | | | Pin # | Particle | GPIO | Peripherals |
|---|---|---|---|---|---|---|---|---|---|
| 3.3V | | | 1 | X | X | 2 | 5V | | |
| I2C | GPIO2 | SDA | 3 | X | X | 4 | 5V | | |
| | GPIO3 | SCL | 5 | X | X | 6 | GND | | |
| Digital I/O | GPIO4 | D0 | 7 | X | X | 8 | TX | GPIO14 | UART |
| GND | | | 9 | X | X | 10 | RX | GPIO15 | Serial 1 |
| Digital I/O | GPIO17 | D1 | 11 | X | X | 12 | D9/A0 | GPIO18 | PWM 1 |
| Digital I/O | GPIO27 | D2 | 13 | X | X | 14 | GND | | |
| Digital I/O | GPIO22 | D3 | 15 | X | X | 16 | D10/A1 | GPIO23 | Digital I/O |
| 3.3V | | | 17 | X | X | 18 | D11/A2 | GPIO24 | Digital I/O |
| SPI | GPIO10 | MOSI | 19 | X | X | 20 | GND | | |
| | GPIO9 | MISO | 21 | X | X | 22 | D12/A3 | GPIO25 | Digital I/O |
| | GPIO11 | SCK | 23 | X | X | 24 | CE0 | GPIO8 | SPI (chip enable) |
| GND | | | 25 | X | X | 26 | CE1 | GPIO7 | |
| DO NOT USE | ID_SD | DO NOT USE | 27 | X | X | 28 | DO NOT USE | ID_SC | DO NOT USE |
| Digital I/O | GPIO5 | D4 | 29 | X | X | 30 | GND | | |
| Digital I/O | GPIO6 | D5 | 31 | X | X | 32 | D13/A4 | GPIO12 | Digital I/O |
| PWM 2 | GPIO13 | D6 | 33 | X | X | 34 | GND | | |
| PWM 2 | GPIO19 | D7 | 35 | X | X | 36 | D14/A5 | GPIO16 | PWM 1 |
| Digital I/O | GPIO26 | D8 | 37 | X | X | 38 | D15/A6 | GPIO20 | Digital I/O |
| GND | | | 39 | X | X | 40 | D16/A7 | GPIO21 | Digital I/O |

### Limitations

- No floating point support.
- Subset of libc because newlib in the toolchain seems to make unaligned memory accesses. At least it does not work.

  How to rebuild the toolchain with -mstrict-align (and possible more)?

### Toolchain

Download Linaro GCC 7.2.1 with newlib and Linaro GCC 7.2.1 for Linux unpack them in `SIMBA_ROOT/...`

### Default system features

The default configuration includes those major features. They are all initialized by `sys_start()` at the startup of the application.

- *Console.*
- *Debug shell.*

### Drivers

Supported drivers for this board.

- *bmp280 — BMP280 temperature and pressure sensor*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *eeprom_i2c — I2C EEPROM*
- *eeprom_soft — Software EEPROM*
- *flash — Flash memory*

- *gnss — Global Navigation Satellite System*
- *hd44780 — Dot matrix LCD*
- *hx711 — HX711 ADC for weigh scales*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *jtag_soft — Software JTAG*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *sht3xd — SHT3x-D Humidity and Temperature Sensor*
- *uart — Universal Asynchronous Receiver/Transmitter*
- *xbee — XBee*
- *xbee_client — XBee client*

### Library Reference

Read more about board specific functionality in the *Xvisor Raspberry Pi 3* module documentation in the Library Reference.

### Memory usage

Below is the memory usage of two applications:

- The minimal-configuration application is configured to only include the bare minimum of functionality for the low level kernel to run. That is, the thread scheduler and system tick.
- The default-configuration application is built with the default configuration, including a lot more functionality. See the list of *Default system features* above for a summary.

| Application | Flash | RAM |
|---|---|---|
| minimal-configuration | 10280 | 3776 |
| default-configuration | 85832 | 38400 |

### Default configuration

Default Standard Library configuration.

| Name | Value |
|---|---|
| CONFIG_ADC | 0 |
| CONFIG_ALIGNMENT | 8 |
| CONFIG_ANALOG_INPUT_PIN | 0 |
| CONFIG_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_ASSERT | 1 |
| CONFIG_ASSERT_FORCE_FATAL | 1 |
| CONFIG_ASSERT_FORCE_PANIC | 0 |
| CONFIG_BMP280 | 1 |

Continued on next page

Table 23 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_BMP280_COVERTION_TIMEOUT_MS | 50 |
| CONFIG_BMP280_DEBUG_LOG_MASK | -1 |
| CONFIG_CAN | 0 |
| CONFIG_CAN_FRAME_TIMESTAMP | 1 |
| CONFIG_CHIPID | 0 |
| CONFIG_CRC_TABLE_LOOKUP | 1 |
| CONFIG_DAC | 0 |
| CONFIG_DEBUG | 1 |
| CONFIG_DHT | 0 |
| CONFIG_DS18B20 | 1 |
| CONFIG_DS18B20_FS_COMMAND_LIST | 1 |
| CONFIG_DS3231 | 1 |
| CONFIG_EEPROM_I2C | 1 |
| CONFIG_EEPROM_I2C_NUMBER_OF_ATTEMPTS | 100 |
| CONFIG_EEPROM_SOFT | 1 |
| CONFIG_EEPROM_SOFT_CRC | CONFIG_EEPROM_SOFT_CRC_32 |
| CONFIG_EEPROM_SOFT_CRC_32 | 0 |
| CONFIG_EEPROM_SOFT_CRC_CCITT | 1 |
| CONFIG_EEPROM_SOFT_SEMAPHORE | 1 |
| CONFIG_EMACS_COLUMNS_MAX | 80 |
| CONFIG_EMACS_HEAP_SIZE | 32768 |
| CONFIG_EMACS_ROWS_MAX | 24 |
| CONFIG_ESP_WIFI | 0 |
| CONFIG_ESP_WIFI_FS_COMMAND_STATUS | 0 |
| CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ | 16000000 |
| CONFIG_EXTI | 0 |
| CONFIG_FAT16 | 1 |
| CONFIG_FATAL_ASSERT | 1 |
| CONFIG_FILESYSTEM_GENERIC | 1 |
| CONFIG_FLASH | 1 |
| CONFIG_FLASH_DEVICE_SEMAPHORE | 1 |
| CONFIG_FLOAT | 0 |
| CONFIG_FS_FS_COMMAND_APPEND | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_COUNTERS_RESET | 1 |
| CONFIG_FS_FS_COMMAND_FILESYSTEMS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_FORMAT | 1 |
| CONFIG_FS_FS_COMMAND_LIST | 1 |
| CONFIG_FS_FS_COMMAND_PARAMETERS_LIST | 1 |
| CONFIG_FS_FS_COMMAND_READ | 1 |
| CONFIG_FS_FS_COMMAND_REMOVE | 1 |
| CONFIG_FS_FS_COMMAND_WRITE | 1 |
| CONFIG_FS_PATH_MAX | 64 |
| CONFIG_GNSS | 1 |
| CONFIG_GNSS_DEBUG_LOG_MASK | -1 |
| CONFIG_HARNESS_BACKTRACE_DEPTH_MAX | 8 |
| CONFIG_HARNESS_DEBUG | 0 |
| CONFIG_HARNESS_EARLY_EXIT | 1 |
| CONFIG_HARNESS_EXPECT_BUFFER_SIZE | 512 |

Continued on next page

Table 23 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_HARNESS_MOCK_ENTRIES_MAX | 1024 |
| CONFIG_HARNESS_SLEEP_MS | 300 |
| CONFIG_HARNESS_WRITE_BACKTRACE_DEPTH_MAX | 0 |
| CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE | 128 |
| CONFIG_HTTP_SERVER_SSL | 0 |
| CONFIG_HX711 | 1 |
| CONFIG_I2C | 1 |
| CONFIG_I2C_FS_COMMAND_READ | 1 |
| CONFIG_I2C_FS_COMMAND_SCAN | 1 |
| CONFIG_I2C_FS_COMMAND_WRITE | 1 |
| CONFIG_I2C_SOFT | 1 |
| CONFIG_JTAG_SOFT | 1 |
| CONFIG_LED_7SEG_HT16K33 | 0 |
| CONFIG_LINUX_SOCKET_DEVICE | 0 |
| CONFIG_LOG_FS_COMMANDS | 1 |
| CONFIG_MCP2515 | 0 |
| CONFIG_MODULE_INIT_ADC | 0 |
| CONFIG_MODULE_INIT_ANALOG_INPUT_PIN | 0 |
| CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN | 0 |
| CONFIG_MODULE_INIT_BUS | 1 |
| CONFIG_MODULE_INIT_CAN | 0 |
| CONFIG_MODULE_INIT_CHAN | 1 |
| CONFIG_MODULE_INIT_CHIPID | 0 |
| CONFIG_MODULE_INIT_DAC | 0 |
| CONFIG_MODULE_INIT_DHT | 0 |
| CONFIG_MODULE_INIT_DS18B20 | 1 |
| CONFIG_MODULE_INIT_DS3231 | 1 |
| CONFIG_MODULE_INIT_ESP_WIFI | 0 |
| CONFIG_MODULE_INIT_EXTI | 0 |
| CONFIG_MODULE_INIT_FLASH | 1 |
| CONFIG_MODULE_INIT_FS | 1 |
| CONFIG_MODULE_INIT_I2C | 1 |
| CONFIG_MODULE_INIT_I2C_SOFT | 1 |
| CONFIG_MODULE_INIT_INET | 0 |
| CONFIG_MODULE_INIT_LOG | 1 |
| CONFIG_MODULE_INIT_MCP2515 | 0 |
| CONFIG_MODULE_INIT_NETWORK_INTERFACE | 0 |
| CONFIG_MODULE_INIT_NRF24L01 | 0 |
| CONFIG_MODULE_INIT_OWI | 1 |
| CONFIG_MODULE_INIT_PIN | 1 |
| CONFIG_MODULE_INIT_PING | 0 |
| CONFIG_MODULE_INIT_POWER | 0 |
| CONFIG_MODULE_INIT_PWM | 0 |
| CONFIG_MODULE_INIT_PWM_SOFT | 0 |
| CONFIG_MODULE_INIT_RANDOM | 0 |
| CONFIG_MODULE_INIT_RE | 1 |
| CONFIG_MODULE_INIT_RWLOCK | 1 |
| CONFIG_MODULE_INIT_SD | 0 |
| CONFIG_MODULE_INIT_SEM | 1 |

Continued on next page

Table 23 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_MODULE_INIT_SETTINGS | 1 |
| CONFIG_MODULE_INIT_SOCKET | 0 |
| CONFIG_MODULE_INIT_SPI | 0 |
| CONFIG_MODULE_INIT_SSL | 0 |
| CONFIG_MODULE_INIT_STD | 1 |
| CONFIG_MODULE_INIT_THRD | 1 |
| CONFIG_MODULE_INIT_TIMER | 1 |
| CONFIG_MODULE_INIT_UART | 1 |
| CONFIG_MODULE_INIT_UART_SOFT | 0 |
| CONFIG_MODULE_INIT_UPGRADE | 0 |
| CONFIG_MODULE_INIT_USB | 0 |
| CONFIG_MODULE_INIT_USB_DEVICE | 0 |
| CONFIG_MODULE_INIT_USB_HOST | 0 |
| CONFIG_MODULE_INIT_WATCHDOG | 0 |
| CONFIG_MONITOR_THREAD | 1 |
| CONFIG_MONITOR_THREAD_PERIOD_US | 2000000 |
| CONFIG_NETWORK_INTERFACE_FS_COMMAND_LIST | 0 |
| CONFIG_NRF24L01 | 0 |
| CONFIG_NVM_EEPROM_SOFT | 1 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE | 16384 |
| CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE | (CONFIG_NVM_SIZE + 8) |
| CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX | 0 |
| CONFIG_NVM_FS_COMMAND_READ | 1 |
| CONFIG_NVM_FS_COMMAND_WRITE | 1 |
| CONFIG_NVM_SIZE | 2040 |
| CONFIG_OWI | 1 |
| CONFIG_PANIC_ASSERT | 1 |
| CONFIG_PANIC_ASSERT_FILE_LINE | 1 |
| CONFIG_PCINT | 0 |
| CONFIG_PIN | 1 |
| CONFIG_PING_FS_COMMAND_PING | 1 |
| CONFIG_PIN_FS_COMMAND_READ | 1 |
| CONFIG_PIN_FS_COMMAND_SET_MODE | 1 |
| CONFIG_PIN_FS_COMMAND_WRITE | 1 |
| CONFIG_POWER | 0 |
| CONFIG_PREEMPTIVE_SCHEDULER | 0 |
| CONFIG_PROFILE_STACK | 1 |
| CONFIG_PWM | 0 |
| CONFIG_PWM_SOFT | 0 |
| CONFIG_RANDOM | 0 |
| CONFIG_RE_DEBUG_LOG_MASK | -1 |
| CONFIG_SD | 0 |
| CONFIG_SERVICE_FS_COMMAND_LIST | 1 |
| CONFIG_SERVICE_FS_COMMAND_START | 1 |
| CONFIG_SERVICE_FS_COMMAND_STOP | 1 |
| CONFIG_SETTINGS_AREA_SIZE | 256 |
| CONFIG_SETTINGS_BLOB | 1 |
| CONFIG_SETTINGS_FS_COMMAND_LIST | 1 |

Continued on next page

Table 23 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SETTINGS_FS_COMMAND_READ | 1 |
| CONFIG_SETTINGS_FS_COMMAND_RESET | 1 |
| CONFIG_SETTINGS_FS_COMMAND_WRITE | 1 |
| CONFIG_SHELL_COMMAND_MAX | 64 |
| CONFIG_SHELL_HISTORY_SIZE | 768 |
| CONFIG_SHELL_MINIMAL | 0 |
| CONFIG_SHELL_PROMPT | "$ " |
| CONFIG_SHT3XD | 1 |
| CONFIG_SOAM_EMBEDDED_DATABASE | 0 |
| CONFIG_SOCKET_RAW | 1 |
| CONFIG_SOFTWARE_I2C | 1 |
| CONFIG_SPC5_BOOT_ENTRY_RCHW | 1 |
| CONFIG_SPC5_RAM_CLEAR_ALL | 1 |
| CONFIG_SPC5_RELOCATE_INIT | 1 |
| CONFIG_SPC5_WATCHDOG_DISABLE | 1 |
| CONFIG_SPI | 0 |
| CONFIG_SPIFFS | 0 |
| CONFIG_START_CONSOLE | CONFIG_START_CONSOLE_UART |
| CONFIG_START_CONSOLE_DEVICE_INDEX | 0 |
| CONFIG_START_CONSOLE_UART_BAUDRATE | 38400 |
| CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE | 32 |
| CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE | 0 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN | 2 |
| CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT | 3 |
| CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION | 1 |
| CONFIG_START_FILESYSTEM | 0 |
| CONFIG_START_FILESYSTEM_ADDRESS | 0 |
| CONFIG_START_FILESYSTEM_FAT16 | 0 |
| CONFIG_START_FILESYSTEM_SIZE | 65536 |
| CONFIG_START_FILESYSTEM_SPIFFS | 1 |
| CONFIG_START_NETWORK | 0 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT | 30 |
| CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD | MyWiFiPassword |
| CONFIG_START_NETWORK_INTERFACE_WIFI_SSID | MyWiFiSSID |
| CONFIG_START_NVM | 0 |
| CONFIG_START_SHELL | 1 |
| CONFIG_START_SHELL_PRIO | 30 |
| CONFIG_START_SHELL_STACK_SIZE | 4096 |
| CONFIG_START_SOAM | 0 |
| CONFIG_START_SOAM_PRIO | 30 |
| CONFIG_START_SOAM_STACK_SIZE | 4096 |
| CONFIG_STD_OUTPUT_BUFFER_MAX | 16 |
| CONFIG_SYSTEM_INTERRUPTS | 1 |
| CONFIG_SYSTEM_INTERRUPT_STACK_SIZE | 0 |
| CONFIG_SYSTEM_TICK_FREQUENCY | 100 |
| CONFIG_SYSTEM_TICK_SOFTWARE | 0 |
| CONFIG_SYS_CONFIG_STRING | 1 |
| CONFIG_SYS_FS_COMMANDS | 1 |
| CONFIG_SYS_LOG_MASK | LOG_UPTO(INFO) |

Continued on next page

Table 23 – continued from previous page

| Name | Value |
|---|---|
| CONFIG_SYS_MEASURE_INTERRUPT_LOAD | 1 |
| CONFIG_SYS_PANIC_BACKTRACE_DEPTH | 24 |
| CONFIG_SYS_PANIC_KICK_WATCHDOG | 0 |
| CONFIG_SYS_RESET_CAUSE | 1 |
| CONFIG_SYS_SIMBA_MAIN_STACK_MAX | 4096 |
| CONFIG_THRD_CPU_USAGE | 1 |
| CONFIG_THRD_DEFAULT_LOG_MASK | LOG_UPTO(INFO) |
| CONFIG_THRD_ENV | 1 |
| CONFIG_THRD_FS_COMMANDS | 1 |
| CONFIG_THRD_IDLE_STACK_SIZE | 2048 |
| CONFIG_THRD_MONITOR_STACK_SIZE | 2048 |
| CONFIG_THRD_SCHEDULED | 1 |
| CONFIG_THRD_STACK_HEAP | 0 |
| CONFIG_THRD_STACK_HEAP_SIZE | 0 |
| CONFIG_THRD_TERMINATE | 1 |
| CONFIG_TIME_UNIX_TIME_TO_DATE | 1 |
| CONFIG_UART | 1 |
| CONFIG_UART_FS_COUNTERS | 1 |
| CONFIG_UART_SOFT | 0 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ENTER | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_ERASE | 1 |
| CONFIG_UPGRADE_FS_COMMAND_APPLICATION_IS_VALID | 1 |
| CONFIG_UPGRADE_FS_COMMAND_BOOTLOADER_ENTER | 1 |
| CONFIG_USB | 0 |
| CONFIG_USB_DEVICE | 0 |
| CONFIG_USB_DEVICE_FS_COMMAND_LIST | 1 |
| CONFIG_USB_DEVICE_PID | 0x8037 |
| CONFIG_USB_DEVICE_VID | 0x2341 |
| CONFIG_USB_HOST | 0 |
| CONFIG_USB_HOST_FS_COMMAND_LIST | 1 |
| CONFIG_WATCHDOG | 0 |
| CONFIG_WS2812 | 0 |
| CONFIG_XBEE | 1 |
| CONFIG_XBEE_CLIENT | 1 |
| CONFIG_XBEE_CLIENT_DEBUG_LOG_MASK | -1 |
| CONFIG_XBEE_CLIENT_RESPONSE_TIMEOUT_MS | 1000 |
| CONFIG_XBEE_DATA_MAX | 120 |

### Homepage

http://xhypervisor.org/

### Mcu

*xvisor_virt_v8*

## 1.5 Examples

Below is a list of simple examples that are useful to understand the basics of *Simba*.

There are a lot more examples and unit tests on Github that shows how to use most of the *Simba* modules.

### 1.5.1 Analog Read

#### About

Read the value of an analog pin periodically once every second and print the read value to standard output.

See the *Analog input pin Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

int main()
{
    int value;
    struct analog_input_pin_t pin;

    sys_start();
    analog_input_pin_module_init();

    /* Initialize the analog input pin. */
```

(continues on next page)

(continued from previous page)

```
    if (analog_input_pin_init(&pin, &pin_a0_dev) != 0) {
        std_printf(FSTR("Failed to initialize the analog input pin.\r\n"));
        return (-1);
    }

    while (1) {
        /* Read the analog pin value and print it. */
        value = analog_input_pin_read(&pin);
        std_printf(FSTR("value = %d\r\n"), value);

        /* Wait 100 ms. */
        thrd_sleep_ms(100);
    }

    return (0);
}
```

The source code can also be found on Github in the examples/analog_read folder.

#### Build and run

Build and run the application.

```
$ cd examples/analog_read
$ make -s BOARD=<board> run
value = 234
value = 249
value = 230
```

### 1.5.2 Analog Write

#### About

Write analog values to an analog output pin to form a sawtooth wave. Connect a LED to the analog output pin and watch the brightness of the LED change.

See the *Analog output pin Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
```

(continues on next page)

(continued from previous page)

```
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

int main()
{
    int value;
    struct analog_output_pin_t pin;

    sys_start();
    analog_output_pin_module_init();

    /* Initialize the analog output pin. */
    analog_output_pin_init(&pin, &pin_d10_dev);

    value = 0;

    while (1) {
        /* Write a sawtooth wave to the analog output pin. */
        analog_output_pin_write(&pin, value);
        value += 5;
        value %= 1024;

        /* Wait ten milliseconds. */
        thrd_sleep_ms(10);
    }

    return (0);
}
```

The source code can also be found on Github in the examples/analog_write folder.

### Build and run

Build and upload the application.

```
$ cd examples/analog_write
$ make -s BOARD=<board> upload
```

### 1.5.3 Blink

#### About

Turn a LED on and off periodically once a second. This example illustrates how to use digital pins and sleep a thread.

See the *Digital pin Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

int main()
{
    struct pin_driver_t led;

    /* Start the system. */
    sys_start();

    /* Initialize the LED pin as output and set its value to 1. */
    pin_init(&led, &pin_led_dev, PIN_OUTPUT);
    pin_write(&led, 1);

    while (1) {
        /* Wait half a second. */
        thrd_sleep_ms(500);

        /* Toggle the LED on/off. */
```

(continues on next page)

(continued from previous page)

```
        pin_toggle(&led);
    }

    return (0);
}
```

The source code can also be found on Github in the examples/blink folder.

#### Build and run

Build and upload the application.

```
$ cd examples/blink
$ make -s BOARD=<board> upload
```

### 1.5.4 DHT

#### About

Read and print the temperature and humidity measured with DHT sensor.

See the *DHT sensor Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2017-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */
```

(continues on next page)

(continued from previous page)

```
#include "simba.h"

int main()
{
    int res;
    struct dht_driver_t dht;
    float temperature;
    float humidty;

    sys_start();

    dht_module_init();

    /* Initialize the DHT driver. */
    dht_init(&dht, &pin_d2_dev);

    while (1) {
        thrd_sleep(2.5);

        /* Read temperature and humidty from the device. */
        res = dht_read(&dht, &temperature, &humidty);

        if (res != 0) {
            std_printf(OSTR("Read failed with %d: %S.\r\n"),
                       res,
                       errno_as_string(res));
            continue;
        }

        /* Print the read temperature and humidty. */
        std_printf(OSTR("Temperature: %f C\r\n"
                        "Humidty:     %f %%RH\r\n"
                        "\r\n"),
                   temperature,
                   humidty);
    }

    return (0);
}
```

The source code can also be found on Github in the examples/dht folder.

### Build and run

Build and run the application.

```
$ cd examples/dht
$ make -s BOARD=<board> run
Temperature: 22.5 C
Humidty:     68 %RH
```

### 1.5.5 DS18B20

#### About

Read and print the temperature measured with one or more DS18B20 sensors.

See the *One-wire temperature sensor Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

int main()
{
    struct owi_driver_t owi;
    struct ds18b20_driver_t ds;
    struct owi_device_t devices[4];
    char temperature[16], *temperature_p;
    int number_of_sensors;
    int i;

    /* Initialization. */
    sys_start();
    ds18b20_module_init();
    owi_init(&owi, &pin_d7_dev, devices, membersof(devices));
    ds18b20_init(&ds, &owi);
    time_busy_wait_us(50000);
```

(continues on next page)

(continued from previous page)

```
    /* Search for devices on the OWI bus. */
    number_of_sensors = owi_search(&owi);
    std_printf(FSTR("Number of sensors: %d\r\n"), number_of_sensors);

    while (1) {
        /* Take a new temperature sample. */
        ds18b20_convert(&ds);

        for (i = 0; i < owi.len; i++) {
            if (devices[i].id[0] != DS18B20_FAMILY_CODE) {
                continue;
            }

            temperature_p = ds18b20_get_temperature_str(&ds,
                                                        devices[i].id,
                                                        temperature);

            if (temperature_p == NULL) {
                temperature_p = "failed to get";
            }

            std_printf(FSTR("Device id: %02x %02x %02x %02x %02x %02x %02x %02x,"
                            " Temperature: %s\r\n"),
                       (unsigned int)devices[i].id[0],
                       (unsigned int)devices[i].id[1],
                       (unsigned int)devices[i].id[2],
                       (unsigned int)devices[i].id[3],
                       (unsigned int)devices[i].id[4],
                       (unsigned int)devices[i].id[5],
                       (unsigned int)devices[i].id[6],
                       (unsigned int)devices[i].id[7],
                       temperature_p);
        }
    }

    return (0);
}
```

The source code can also be found on Github in the examples/ds18b20 folder.

### Build and run

Build and run the application.

```
$ cd examples/ds18b20
$ make -s BOARD=<board> run
Number of sensors: 2
Device id: 28 9c 1d 5d 05 00 00 32, Temperature: 22.6250
Device id: 28 95 32 5d 05 00 00 33, Temperature: 22.6875
```

### 1.5.6 Filesystem

#### About

Create the file `counter.txt` and write `0` to it. Everytime the application is restarted the counter is incremented by one.

See the *Debug and virtual file system Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

#if !defined(BOARD_ARDUINO_DUE) && !defined(ARCH_ESP) && !defined(ARCH_ESP32)
#    error "This example can only be built for Arduino Due, ESP and ESP32."
#endif

/**
 * Increment the counter in 'counter.txt'.
 */
static int increment_counter(void)
{
    char buf[32];
    struct fs_file_t file;
    long counter;
    size_t size;

    std_printf(FSTR("Incrementing the counter in 'counter.txt'.\r\n"));
```

(continues on next page)

(continued from previous page)

```
    if (fs_open(&file, "counter.txt", FS_RDWR) != 0) {
        /* Create the file if missing. */
        if (fs_open(&file,
                    "counter.txt",
                    FS_CREAT | FS_TRUNC | FS_RDWR) != 0) {
            return (-1);
        }

        if (fs_write(&file, "0", 2) != 2) {
            return (-2);
        }

        if (fs_seek(&file, 0, FS_SEEK_SET) != 0) {
            return (-3);
        }
    }

    if (fs_read(&file, buf, 16) <= 0) {
        return (-4);
    }

    if (std_strtol(buf, &counter) == NULL) {
        return (-5);
    }

    /* Increment the counter. */
    counter++;
    std_sprintf(buf, FSTR("%lu"), counter);
    size = strlen(buf) + 1;

    if (fs_seek(&file, 0, FS_SEEK_SET) != 0) {
        return (-6);
    }

    if (fs_write(&file, buf, size) != size) {
        return (-7);
    }

    if (fs_close(&file) != 0) {
        return (-8);
    }

    std_printf(FSTR("Counter incremented to %lu\r\n"), counter);

    return (0);
}

int main()
{
    int res;

    sys_start();
    std_printf(sys_get_info());

    /* Increment the counter. */
    res = increment_counter();
```

(continues on next page)

(continued from previous page)

```
    if (res != 0) {
        std_printf(FSTR("Failed to increment the counter with error %d.\r\n"),
                   res);
    }

    /* The shell thread is started in sys_start() so just suspend this
       thread. */
    thrd_suspend(NULL);

    return (0);
}
```

The source code can also be found on Github in the examples/filesystem folder.

#### Build and run

Build and run the application.

```
$ cd examples/filesystem
$ make -s BOARD=arduino_due upload
```

The output in the terminal emulator:

```
Incrementing the counter in 'counter.txt'.
Counter incremented to 1.
<manually reset the board>
Incrementing the counter in 'counter.txt'.
Counter incremented to 2.
<manually reset the board>
Incrementing the counter in 'counter.txt'.
Counter incremented to 3.
```

### 1.5.7 Hello World

#### About

This application prints "Hello world!" to standard output.

See the *Standard functions Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
```

(continues on next page)

(continued from previous page)

```
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

int main()
{
    /* Start the system. */
    sys_start();

    std_printf(OSTR("Hello world!\r\n"));

    return (0);
}
```

The source code can also be found on Github in the examples/hello_world folder.

### Build and run

Build and run the application.

```
$ cd examples/hello_world
$ make -s BOARD=<board> run
...
Hello world!
$
```

## 1.5.8 HTTP Client

### About

Conenct to a remote host perform a HTTP GET action to fetch the root page '/' from the remote host.

See the *Socket Library Reference* for more details.

### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

/* The ip address of the host to connect to. */
#define REMOTE_HOST_IP 216.58.211.142

int main()
{
    struct socket_t socket;
    char http_request[] =
        "GET / HTTP/1.1\r\n"
        "Host: " STRINGIFY(REMOTE_HOST_IP) "\r\n"
        "\r\n";
    char http_response[64];
    char remote_host_ip[] = STRINGIFY(REMOTE_HOST_IP);
    struct inet_addr_t remote_host_address;

    /* Start the system. Brings up the configured network interfaces
       and starts the TCP/IP-stack. */
    sys_start();

    /* Open the tcp socket. */
    socket_open_tcp(&socket);

    std_printf(FSTR("Connecting to '%s'.\r\n"), remote_host_ip);

    if (inet_aton(remote_host_ip, &remote_host_address.ip) != 0) {
        std_printf(FSTR("Bad ip address ''.\r\n"), remote_host_ip);
```

(continues on next page)

(continued from previous page)

```
        return (-1);
    }

    remote_host_address.port = 80;

    if (socket_connect(&socket, &remote_host_address) != 0) {
        std_printf(FSTR("Failed to connect to '%s'.\r\n"), remote_host_ip);
        return (-1);
    }

    /* Send the HTTP request... */
    if (socket_write(&socket,
                     http_request,
                     strlen(http_request)) != strlen(http_request)) {
        std_printf(FSTR("Failed to send the HTTP request.\r\n"));
        return (-1);
    }

    /* ...and receive the first 64 bytes of the response. */
    if (socket_read(&socket,
                    http_response,
                    sizeof(http_response)) != sizeof(http_response)) {
        std_printf(FSTR("Failed to receive the response.\r\n"));
    }

    std_printf(FSTR("First 64 bytes of the response:\r\n"
                    "%s"),
               http_response);

    /* Close the socket. */
    socket_close(&socket);

    return (0);
}
```

The source code can also be found on Github in the examples/http_client folder.

### Build and run

Build and run the application.

```
$ cd examples/http_client
$ make -s BOARD=<board> SSID=<wifi SSID> PASSWORD=<wifi password> run
...
Connecting to WiFi with SSID 'Qvist'.
Connected to WiFi with SSID 'Qvist'. Got IP address '192.168.1.103'.
Connecting to '216.58.211.142'.
First 64 bytes of the response:
HTTP/1.1 301 Moved Permanently
Location: http://www.google.com/GET / HTTP/1.1
Host: 216.58.211.142
...
$
```

### 1.5.9 Ping

#### About

Ping a remote host periodically once every second.

See the *Ping Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

/* The ip address of the host to ping. */
#define REMOTE_HOST_IP 8.8.4.4

int main()
{
    int res, attempt;
    char remote_host_ip[] = STRINGIFY(REMOTE_HOST_IP);
    struct inet_ip_addr_t remote_host_ip_address;
    struct time_t round_trip_time, timeout;

    sys_start();

    if (inet_aton(remote_host_ip, &remote_host_ip_address) != 0) {
        std_printf(FSTR("Bad ip address '%s'.\r\n"), remote_host_ip);
        return (-1);
    }
```

(continues on next page)

(continued from previous page)

```
    timeout.seconds = 3;
    timeout.nanoseconds = 0;
    attempt = 1;

    /* Ping the remote host once every second. */
    while (1) {
        res = ping_host_by_ip_address(&remote_host_ip_address,
                                      &timeout,
                                      &round_trip_time);

        if (res == 0) {
            std_printf(FSTR("Successfully pinged '%s' (#%d).\r\n"),
                       remote_host_ip,
                       attempt);
        } else {
            std_printf(FSTR("Failed to ping '%s' (#%d).\r\n"),
                       remote_host_ip,
                       attempt);
        }

        attempt++;
        thrd_sleep(1);
    }

    return (0);
}
```

The source code can also be found on Github in the examples/ping folder.

### Build and run

Build and run the application.

```
$ cd examples/ping
$ make -s BOARD=<board> SSID=<wifi SSID> PASSWORD=<wifi password> run
Successfully pinged '8.8.4.4' in 20 ms (#1).
Successfully pinged '8.8.4.4' in 20 ms (#2).
Successfully pinged '8.8.4.4' in 20 ms (#3).
```

## 1.5.10 Queue

### About

Use a queue to communicate between two threads.

See the *Queue Library Reference* for more details.

### Source code

```
/*
 * The MIT License (MIT)
```

(continues on next page)

(continued from previous page)

```
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

static struct queue_t queue;

static THRD_STACK(writer_stack, 256);

static void *writer_main(void *arg_p)
{
    int value;

    /* Write to the queue. */
    value = 1;
    queue_write(&queue, &value, sizeof(value));

    return (NULL);
}

int main()
{
    int value;

    sys_start();
    queue_init(&queue, NULL, 0);
    thrd_spawn(writer_main, NULL, 0, writer_stack, sizeof(writer_stack));

    /* Read from the queue. */
    queue_read(&queue, &value, sizeof(value));

    std_printf(FSTR("read value = %d\r\n"), value);

    return (0);
```

(continues on next page)

(continued from previous page)

```
}
```

The source code can also be found on Github in the examples/queue folder.

#### Build and run

Build and upload the application.

```
$ cd examples/queue
$ make -s BOARD=<board> run
read value = 1
```

### 1.5.11 Shell

#### About

Use the serial port to monitor and control the application.

See the *Debug shell Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"
```

(continues on next page)

(continued from previous page)

```
/* Hello world command. */
static struct fs_command_t cmd_hello_world;

static struct shell_t shell;

/**
 * The shell command callback for "/hello_world".
 */
static int cmd_hello_world_cb(int argc,
                              const char *argv[],
                              void *out_p,
                              void *in_p,
                              void *arg_p,
                              void *call_arg_p)
{
    /* Write "Hello World!" to the output channel. */
    std_fprintf(out_p, OSTR("Hello World!\r\n"));

    return (0);
}

int main()
{
    /* Start the system. */
    sys_start();

    std_printf(sys_get_info());

    /* Register the hello world command. */
    fs_command_init(&cmd_hello_world,
                    CSTR("/hello_world"),
                    cmd_hello_world_cb,
                    NULL);
    fs_command_register(&cmd_hello_world);

    /* Start the shell. */
    shell_init(&shell,
               sys_get_stdin(),
               sys_get_stdout(),
               NULL,
               NULL,
               NULL,
               NULL);
    shell_main(&shell);

    return (0);
}
```

The source code can also be found on Github in the examples/shell folder.

### Build and run

Build and run the application.

```
$ cd examples/shell
$ make -s BOARD=<board> upload
```

Communicate with the board using a serial terminal emulator, for example *TeraTerm*.

Type `hello_world` in the terminal emulator and press Enter. `Hello World!` is printed.

Press Tab to print a list of all registered commands and try them if you want to.

```
$ hello_world
Hello World!
$ <tab>
drivers/
filesystems/
hello_world
help
history
kernel/
logout
oam/
$ kernel/thrd/list
            NAME        STATE  PRIO   CPU  MAX-STACK-USAGE  LOGMASK
           shell      current     0    0%       358/  5575     0x0f
            idle        ready   127    0%        57/   156     0x0f
$
```

### 1.5.12 Timer

#### About

Start a periodic timer that writes an event to the main thread. The main thread reads the event and prints "timeout" to the standard output.

See the *Timer Library Reference* for more details.

#### Source code

```
/*
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2018, Erik Moqvist
 *
 * Permission is hereby granted, free of charge, to any person
 * obtaining a copy of this software and associated documentation
 * files (the "Software"), to deal in the Software without
 * restriction, including without limitation the rights to use, copy,
 * modify, merge, publish, distribute, sublicense, and/or sell copies
 * of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
 * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
```

(continues on next page)

(continued from previous page)

```
 * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * This file is part of the Simba project.
 */

#include "simba.h"

#define TIMEOUT_EVENT    0x1

static struct event_t event;
static struct timer_t timer;

static void timer_cb(void *arg_p)
{
    uint32_t mask;

    mask = TIMEOUT_EVENT;
    event_write_isr(&event, &mask, sizeof(mask));
}

int main()
{
    uint32_t mask;
    struct time_t timeout;

    sys_start();
    event_init(&event);

    /* Initialize and start a periodic timer. */
    timeout.seconds = 1;
    timeout.nanoseconds = 0;
    timer_init(&timer, &timeout, timer_cb, NULL, TIMER_PERIODIC);
    timer_start(&timer);

    while (1) {
        mask = TIMEOUT_EVENT;
        event_read(&event, &mask, sizeof(mask));

        std_printf(FSTR("timeout\r\n"));
    }

    return (0);
}
```

The source code can also be found on Github in the examples/timer folder.

### Build and run

Build and upload the application.

```
$ cd examples/timer
$ make -s BOARD=<board> run
timeout
```

(continues on next page)

(continued from previous page)

```
timeout
timeout
```

## 1.6 Library Reference

Simba's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains modules used by many developers in their everyday programming.

Besides the generated documentation, the source code of the interfaces and their implementatins are available on Github.

### 1.6.1 kernel

The kernel package is the heart in *Simba*. It implements the thread scheduler.

The kernel package on Github.

#### `assert` — Assertions

An assertion tests a condition, and either returns an error or stops the application.

There are three kind of assertions in Simba; `ASSERT()`, `FATAL_ASSERT()` and `PANIC_ASSERT()`.

- `ASSERT()` is configurable to either return an error code, or call the fatal callback.
- `FATAL_ASSERT()` always calls the fatal callback.
- `PANIC_ASSERT()` calls the panic function, which allows this assertion to be used in interrupt context.

#### Example usage

All assertion macros share the same prototype, in this example testing one condition each.

```
ASSERT(x > 1);

FATAL_ASSERT(y != 2);

PANIC_ASSERT(z < 3);
```

---

Source code: src/kernel/assert.h

---

#### Defines

**FATAL**(n)
Make an assert fatal by negating the error code.

**IS_FATAL**(n)
Check is an error code is fatal (negative error code).

**ASSERT**(cond, ...)
Assert given condition and print an error message on assertion failure. Call the system on fatal callback with error code `EASSERT` on fatal error, otherwise return NULL.

**ASSERTN**(cond, n, ...)
Assert given condition and print an error message on assertion failure. Call the system on fatal callback with given error code `n` on fatal error, otherwise return the error code negated.

**ASSERTRV**(cond, ...)
Assert given condition and print an error message on assertion failure. Call the system on fatal callback with error code `EASSERT` on fatal error, otherwise return NULL.

**ASSERTRN**(cond, ...)
Assert given condition and print an error message on assertion failure. Call the system on fatal callback with error code `EASSERT` on fatal error, otherwise return NULL.

**ASSERTNR**(cond, n, ...)
Assert given condition and print an error message. Call the system on fatal callback with given error code `n` on fatal error, otherwise return given error code `res`.

**ASSERTNRV**(cond, n, ...)
Assert given condition and print an error message on assertion failure. Call the system on fatal callback with given error code `n` on fatal error, otherwise return.

**ASSERTNRN**(cond, n, ...)
Assert given condition and print an error message on assertion failure. Call the system on fatal callback with given error code `n` on fatal error, otherwise return NULL.

**FATAL_ASSERTN**(cond, n, ...)
Assert given condition and print an error message on assertion failure, then call the system on fatal callback with given error code `n`.

This assertion is not affected by `CONFIG_ASSERT`, but instead `CONFIG_FATAL_ASSERT`.

**FATAL_ASSERT**(cond, ...)
Assert given condition and print an error message on assertion failure, then call the system on fatal callback with error code `EASSERT`.

This assertion is not affected by `CONFIG_ASSERT`, but instead `CONFIG_FATAL_ASSERT`.

**PANIC_ASSERTN**(cond, n, ...)
Assert given condition and call `sys_panic()` with given error code `n` on assertion failure.

This assertion is not affected by `CONFIG_ASSERT`, but instead `CONFIG_PANIC_ASSERT`.

**PANIC_ASSERT**(cond, ...)
Assert given condition and call `sys_panic()` with error code `EASSERT`.

This assertion is not affected by `CONFIG_ASSERT`, but instead `CONFIG_PANIC_ASSERT`.

### errno — Error numbers

Error numbers (or codes) are often returned from a function when an error occurred.

---

Source code: src/kernel/errno.h

---

### Defines

**EPERM**
Operation not permitted.

**ENOENT**
No such file or directory.

**ESRCH**
No such process.

**EINTR**
Interrupted system call.

**EIO**
I/O error.

**ENXIO**
No such device or address.

**E2BIG**
Argument list too long.

**ENOEXEC**
Exec format error.

**EBADF**
Bad file number.

**ECHILD**
No child processes.

**EAGAIN**
Try again.

**ENOMEM**
Out of memory.

**EACCES**
Permission denied.

**EFAULT**
Bad address.

**ENOTBLK**
Block device required.

**EBUSY**
Device or resource busy.

**EEXIST**
File exists.

**EXDEV**
Cross-device link.

**ENODEV**
No such device.

**ENOTDIR**
Not a directory.

**EISDIR**
Is a directory.

**EINVAL**
Invalid argument.

**ENFILE**
File table overflow.

**EMFILE**
Too many open files.

**ENOTTY**
Not a typewriter.

**ETXTBSY**
Text file busy.

**EFBIG**
File too large.

**ENOSPC**
No space left on device.

**ESPIPE**
Illegal seek.

**EROFS**
Read-only file system.

**EMLINK**
Too many links.

**EPIPE**
Broken pipe.

**EDOM**
Math argument out of domain of func.

**ERANGE**
Math result not representable.

**EDEADLK**
Resource deadlock would occur.

**ENAMETOOLONG**
File name too long.

**ENOLCK**
No record locks available.

**ENOSYS**
Function not implemented.

**ENOTEMPTY**
Directory not empty.

**ELOOP**
Too many symbolic links encountered.

**EWOULDBLOCK**
Operation would block.

**ENOMSG**
No message of desired type.

**EIDRM**
Identifier removed.

**ECHRNG**
Channel number out of range.

**EL2NSYNC**
Level 2 not synchronized.

**EL3HLT**
Level 3 halted.

**EL3RST**
Level 3 reset.

**ELNRNG**
Link number out of range.

**EUNATCH**
Protocol driver not attached.

**ENOCSI**
No CSI structure available.

**EL2HLT**
Level 2 halted.

**EBADE**
Invalid exchange.

**EBADR**
Invalid request descriptor.

**EXFULL**
Exchange full.

**ENOANO**
No anode.

**EBADRQC**
Invalid request code.

**EBADSLT**
Invalid slot.

**EDEADLOCK**
Deadlock.

**EBFONT**
Bad font file format.

**ENOSTR**
Device not a stream.

**ENODATA**
No data available.

**ETIME**
Timer expired.

**ENOSR**
Out of streams resources.

**ENONET**
Machine is not on the network.

**ENOPKG**
Package not installed.

**EREMOTE**
Object is remote.

**ENOLINK**
Link has been severed.

**EADV**
Advertise error.

**ESRMNT**
Srmount error.

**ECOMM**
Communication error on send.

**EPROTO**
Protocol error.

**EMULTIHOP**
Multihop attempted.

**EDOTDOT**
RFS specific error.

**EBADMSG**
Not a data message.

**EOVERFLOW**
Value too large for defined data type.

**ENOTUNIQ**
Name not unique on network.

**EBADFD**
File descriptor in bad state.

**EREMCHG**
Remote address changed.

**ELIBACC**
Can not access a needed shared library.

**ELIBBAD**
Accessing a corrupted shared library.

**ELIBSCN**
.lib section in a.out corrupted.

**ELIBMAX**
Attempting to link in too many shared libraries.

**ELIBEXEC**
Cannot exec a shared library directly.

**EILSEQ**
Illegal byte sequence.

**ERESTART**
Interrupted system call should be restarted.

**ESTRPIPE**
Streams pipe error.

**EUSERS**
Too many users.

**ENOTSOCK**
Socket operation on non-socket.

**EDESTADDRREQ**
Destination address required.

**EMSGSIZE**
Message too long.

**EPROTOTYPE**
Protocol wrong type for socket.

**ENOPROTOOPT**
Protocol not available.

**EPROTONOSUPBOARD**
Protocol not supported.

**ESOCKTNOSUPBOARD**
Socket type not supported.

**EOPNOTSUPP**
Operation not supported on transport endpoint.

**EPFNOSUPBOARD**
Protocol family not supported.

**EAFNOSUPBOARD**
Address family not supported by protocol.

**EADDRINUSE**
Address already in use.

**EADDRNOTAVAIL**
Cannot assign requested address.

**ENETDOWN**
Network is down.

**ENETUNREACH**
Network is unreachable.

**ENETRESET**
Network dropped connection because of reset.

**ECONNABORTED**
Software caused connection abort.

**ECONNRESET**
Connection reset by peer.

**ENOBUFS**
No buffer space available.

**EISCONN**
Transport endpoint is already connected.

**ENOTCONN**
Transport endpoint is not connected.

**ESHUTDOWN**
Cannot send after transport endpoint shutdown.

**ETOOMANYREFS**
Too many references: cannot splice.

**ETIMEDOUT**
Connection timed out.

**ECONNREFUSED**
Connection refused.

**EHOSTDOWN**
Host is down.

**EHOSTUNREACH**
No route to host.

**EALREADY**
Operation already in progress.

**EINPROGRESS**
Operation now in progress.

**ESTALE**
Stale NFS file handle.

**EUCLEAN**
Structure needs cleaning.

**ENOTNAM**
Not a XENIX named type file.

**ENAVAIL**
No XENIX sems available.

**EISNAM**
Is a named type file.

**EREMOTEIO**
Remote I/O error.

**EDQUOT**
Quota exceeded.

**ENOMEDIUM**
No medium found.

**EMEDIUMTYPE**
Wrong medium type.

**ECANCELED**
Operation Canceled.

**ENOKEY**
Required key not available.

**EKEYEXPIRED**
Key has expired.

**EKEYREVOKED**
Key has been revoked.

**EKEYREJECTED**
Key was rejected by service.

**ESTACK**
Stack corrupt.

**EBTASSERT**
Test assertion.

**EASSERT**
Assertion.

**ENOCOMMAND**
Command not found.

**ENOTMOUNTED**
Resource not mounted.

**EKEYNOTFOUND**
Key not found.

**EBADVALUE**
Bad value.

**EBADCRC**
Bad CRC.

**EFLASHWRITE**
Flash write failed.

**EFLASHERASE**
Flash erase failed.

### Functions

far_string_t **errno_as_string**(int *errno*)
Map given error number to a string.

**Return** Error number as a far string or NULL if it's not an error number.

**Parameters**

- `errno`: Error number to map to a string. Both positive and negative error numbers are accepted.

## sys — System

System level functionality and definitions.

### Example usage

This is a small example illustrating how to start an application by calling *sys_start()* and then print the system info and uptime on standard output.

```
int main()
{
    struct time_t uptime;

    sys_start();

    /* Print the system information. */
    std_printf(sys_get_info());

    /* Get the system uptime and print it. */
    sys_uptime(&uptime)
    std_printf(OSTR("System uptime: %lu s %lu ns"),
               uptime.seconds,
               uptime.nanoseconds);

    return (0);
}
```

### Debug file system commands

Seven debug file system commands are available, all located in the directory `kernel/sys/`.

| Command | Description |
|---|---|
| `info` | Print the system information. |
| `config` | Print the build configuration (or at least part of it). |
| `uptime` | Print the system uptime. |
| `panic` | Call the panic function, stopping the system. |
| `reboot` | Reboot the system. |
| `backtrace` | Print a backtrace. |
| `reset_cause` | Print the reset cause. |

Example output from the shell:

```
$ kenel/sys/info
app:    shell-master built 2017-07-06 09:03 CEST by erik.
board:  Arduino Mega
mcu:    Atmel ATMega2560 AVR @ 16MHz, 8k sram, 256k flash
OK
$ kenel/sys/uptime
0.120 seconds
OK
$ kenel/sys/reset_cause
power_on
OK
```

---

Source code: src/kernel/sys.h, src/kernel/sys.c

Test code: tst/kernel/sys/main.c

Test coverage: src/kernel/sys.c

---

### Defines

**VERSION_STR**

**SYS_TICK_MAX**

### Typedefs

**typedef** uint32_t **sys_tick_t**

**typedef** uint32_t **cpu_usage_t**

**typedef** void (***sys_on_fatal_fn_t**) (int error)

### Enums

**enum sys_reset_cause_t**

System reset causes.

*Values:*

**sys_reset_cause_unknown_t** = 0

**sys_reset_cause_power_on_t**

**sys_reset_cause_watchdog_timeout_t**

**sys_reset_cause_software_t**

**sys_reset_cause_external_t**

**sys_reset_cause_jtag_t**

**sys_reset_cause_max_t**

### Functions

**static** *sys_tick_t* **t2st** (**const struct** *time_t* **time_p*)

Convertion from the time struct to system ticks.

**static** void **st2t** (*sys_tick_t* *tick*, **struct** *time_t* **time_p*)

Convertion from system ticks to the time struct.

int **sys_module_init** (void)

Initialize the sys module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **sys_start** (void)

Start the system and convert this context to the main thread.

This function initializes a bunch of enabled features in the simba platform. Many low level features (scheduling, timers, ...) are always enabled, but higher level features are only enabled if configured.

This function **must** be the first function call in main().

**Return** zero(0) or negative error code.

void **sys_stop**(int *error*)
Stop the system.

**Return** Never returns.

**Parameters**

- `error`: Error code.

void **sys_panic**(far_string_t *fmt_p*, ...)
System panic. Write given message, a backtrace and other port specific debug information to the console and then reboot the system.

This function may be called from interrupt context and with the system lock taken.

**Return** Never returns.

**Parameters**

- `fmt_p`: Format string.
- `...`: Variable arguments list.

void **sys_reboot**(void)
Reboot the system. Also known as a soft reset.

**Return** Never returns.

int **sys_backtrace**(void ****buf_p*, size_t *size*)
Store the backtrace in given buffer.

**Return** Backtrace depth.

**Parameters**

- `buf_p`: Buffer to store the backtrace in.
- `size`: Size of the buffer.

**enum** *sys_reset_cause_t* **sys_reset_cause**(void)
Get the system reset cause.

**Return** The reset cause.

int **sys_uptime**(**struct** *time_t* **uptime_p*)
Get the system uptime.

**Return** zero(0) or negative error code.

**Parameters**

- `uptime_p`: System uptime.

int **sys_uptime_isr**(**struct** *time_t* **uptime_p*)
Get the system uptime from interrupt context or with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `uptime_p`: System uptime.

void **sys_set_on_fatal_callback**(*sys_on_fatal_fn_t callback*)
Set the on-fatal-callback function to given callback.

The on-fatal-callback is called when a fatal error occurs. The default on-fatal-callback is `sys_stop()`.

**Return** void

**Parameters**

- `callback`: Callback called when a fatal error occurs.

void **sys_set_stdin**(void *chan_p)
Set the standard input channel.

**Return** void.

**Parameters**

- `chan_p`: New standard input channel.

void ***sys_get_stdin**(void)
Get the standard input channel.

**Return** Standard input channel.

void **sys_set_stdout**(void *chan_p)
Set the standard output channel.

**Return** void.

**Parameters**

- `chan_p`: New standard output channel.

void ***sys_get_stdout**(void)
Get the standard output channel.

**Return** Standard output channel.

void **sys_lock**(void)
Take the system lock. Turns off interrupts.

**Return** void.

void **sys_unlock**(void)
Release the system lock. Turn on interrupts.

**Return** void.

void **sys_lock_isr**(void)
Take the system lock from isr. In many ports this has no effect.

**Return** void.

void **sys_unlock_isr**(void)
Release the system lock from isr. In many ports this function has no effect.

**Return** void.

far_string_t **sys_get_info**(void)
Get a pointer to the application information string.

The buffer contains various information about the application; for example the application name and the build date.

**Return** The pointer to the application information string.

far_string_t **sys_get_config**(void)
Get a pointer to the application configuration string.

The buffer contains a string of all configuration variables and their values.

**Return** The pointer to the application configuration string.

*cpu_usage_t* **sys_interrupt_cpu_usage_get**(void)
Get the current interrupt cpu usage counter.

**Return** cpu usage, 0-100.

void **sys_interrupt_cpu_usage_reset**(void)
Reset the interrupt cpu usage counter.

far_string_t **sys_reset_cause_as_string**(**enum** *sys_reset_cause_t reset_cause*)
Get the reset cause as a far string.

### Variables

**struct** *sys_t* **sys**

**struct sys_t**

#### Public Members

*sys_on_fatal_fn_t* **on_fatal_callback**

void ***stdin_p**

void ***stdout_p**

## thrd — Threads

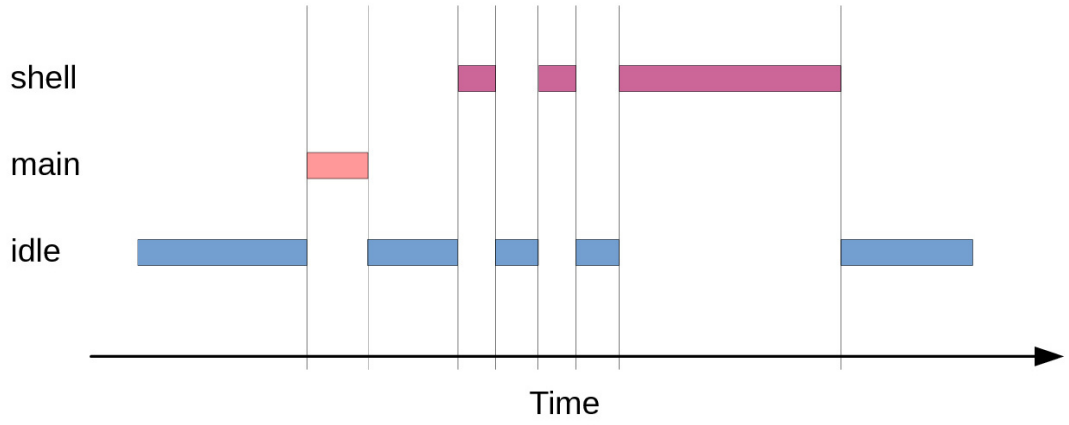
A thread is the basic execution entity in the OS. A pre-emptive or cooperative scheduler controls the execution of threads.

### Scheduler

The single core scheduler is configured as cooperative or preemptive at compile time. The cooperative scheduler is implemented for all boards, but the preemptive scheduler is only implemented for a few boards.

There are two threads that are always present; the main thread and the idle thread. The main thread is the root thread in the system, created in the `main()` function by calling `sys_start()`. The idle thread is running when no other thread is ready to run. It simply waits for an interrupt to occur and then reschedules to run other ready threads.

The diagram below is an example of how three threads; `shell`, `main` and `idle` are scheduled over time.




As it is a single core scheduler only one thread is runnng at a time. In the beginning the system is idle and the `idle` thread is running. After a while the `main` and `shell` threads have some work to do, and since they have higher priority than the `idle` thread they are scheduled. At the end the `idle` thread is running again.

### Debug file system commands

Four debug file system commands are available, all located in the directory `kernel/thrd/`.

| Command | Description |
|---|---|
| `list` | Print a list of all threads. |
| `set_log_mask <thread name> <mask>` | Set the log mask of thread `<thread name>` to `mask`. |
| `monitor/set_period_ms <ms>` | Set the monitor thread sampling period to `<ms>` milliseconds. |
| `monitor/set_print <state>` | Enable(`1`)/disable(`0`) monitor statistics to be printed periodically. |

Example output from the shell:

```
$ kenel/thrd/list
            NAME         STATE  PRIO   CPU   SCHEDULED  LOGMASK
            main       current     0    0%           1     0x0f
                         ready   127    0%           0     0x0f
                         ready   -80    0%           0     0x0f
OK
```

---

Source code: src/kernel/thrd.h, src/kernel/thrd.c

Test code: tst/kernel/thrd/main.c

Test coverage: src/kernel/thrd.c

## Defines

**THRD_STACK**(name, size)
Macro to declare a thread stack with given name and size. All thread stacks must be defined using this macro.

**Parameters**

- `name`: The name of the stack. A variable is declared with this name that should be passed to `thrd_spawn()`.
- `size`: Size of the stack in bytes.

**THRD_CONTEXT_STORE_ISR**
Push all callee-save registers not part of the context struct. The preemptive scheduler requires this macro before the `thrd_yield_isr()` function is called from interrupt context.

**THRD_CONTEXT_LOAD_ISR**
Pop all callee-save registers not part of the context struct. The preemptive scheduler requires this macro after the thrd_yield_isr() function is called from interrupt context.

**THRD_RESCHEDULE_ISR**
Reschuedule from isr. Used by preemptive systems to interrupt low priority threads in favour of high priority threads.

## Functions

int **thrd_module_init**(void)
Initialize the thread module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code

**struct** *thrd_t* ***thrd_spawn**(void *(**main*)) void *
, void **arg_p*, int *prio*, void **stack_p*, size_t *stack_size*Spawn a thread with given main (entry) function and argument. The thread is initialized and added to the ready queue in the scheduler for execution when prioritized.

**Return** Thread id, or NULL on error.

**Parameters**

- `main`: Thread main (entry) function. This function normally contains an infinate loop waiting for events to occur.
- `arg_p`: Main function argument. Passed as arg_p to the main function.
- `prio`: Thread scheduling priority. [-127..127], where -127 is the highest priority and 127 is the lowest.
- `stack_p`: Stack pointer. The pointer to a stack created with the macro `THRD_STACK()`.
- `stack_size`: The stack size in number of bytes.

int **thrd_suspend**(**const struct** *time_t *timeout_p*)
Suspend current thread and wait to be resumed or a timeout occurs (if given).

**Return** zero(0), -ETIMEDOUT on timeout or other negative error code.

**Parameters**

- `timeout_p`: Time to wait to be resumed before a timeout occurs and the function returns.

int **thrd_resume**(**struct** *thrd_t *thrd_p*, int *err*)
Resume given thread. If resumed thread is not yet suspended it will not be suspended on next suspend call to `thrd_suspend()` or `thrd_suspend_isr()`.

**Return** zero(0) or negative error code.

**Parameters**

- `thrd_p`: Thread id to resume.
- `err`: Error code to be returned by `thrd_suspend()` or `thrd_suspend_isr()`.

int **thrd_yield**(void)
Put the currently executing thread on the ready list and reschedule.

This function is often called periodically from low priority work heavy threads to give higher priority threads the chance to execute.

**Return** zero(0) or negative error code.

int **thrd_join**(**struct** *thrd_t *thrd_p*)
Wait for given thread to terminate.

**Return** zero(0) or negative error code.

**Parameters**

- `thrd_p`: Thread to wait for.

int **thrd_terminate**(**struct** *thrd_t *thrd_p*)
Terminate given thread. The stack of a terminated thread may *not* be reused in the current implementation.

**Return** zero(0) or negative error code.

**Parameters**

- `thrd_p`: Thread to terminate.

int **thrd_sleep**(float *seconds*)
Pauses the current thread for given number of seconds.

**Return** zero(0) or negative error code.

**Parameters**

- `seconds`: Seconds to sleep.

int **thrd_sleep_ms**(int *milliseconds*)
Pauses the current thread for given number of milliseconds.

**Return** zero(0) or negative error code.

**Parameters**

- `milliseconds`: Milliseconds to sleep.

int **thrd_sleep_us**(long *microseconds*)
Pauses the current thread for given number of microseconds.

**Return** zero(0) or negative error code.

**Parameters**

- `microseconds`: Microseconds to sleep.

**struct** *thrd_t* ***thrd_self**(void)
Get current thread's id.

**Return** Thread id.

int **thrd_set_name**(**const** char **name_p*)
Set the name of the current thread.

**Return** zero(0) or negative error code.

**Parameters**

- `name_p`: New thread name.

**const** char ***thrd_get_name**(void)
Get the name of the current thread.

**Return** Current thread name.

**struct** *thrd_t* ***thrd_get_by_name**(**const** char **name_p*)
Get the pointer to given thread.

**Return** Thraed pointer or NULL if the thread was not found.

int **thrd_set_log_mask**(**struct** *thrd_t* **thrd_p*, int *mask*)
Set the log mask of given thread.

**Return** Old log mask.

**Parameters**

- `thrd_p`: Thread to set the log mask of.
- `mask`: Log mask. See the log module for available levels.

int **thrd_get_log_mask**(void)
Get the log mask of the current thread.

**Return** Log mask of current thread.

int **thrd_set_prio**(**struct** *thrd_t* **thrd_p*, int *prio*)
Set the priority of given thread.

**Return** zero(0) or negative error code.

**Parameters**

- `thrd_p`: Thread to set the priority for.
- `prio`: Priority.

int **thrd_get_prio**(void)
Get the priority of the current thread.

**Return** Priority of current thread.

int **thrd_init_global_env**(**struct** *thrd_environment_variable_t* *variables_p, int *length*)
Initialize the global environment variables storage. These variables are shared among all threads.

**Return** zero(0) or negative error code.

**Parameters**

- `variables_p`: Variables array.
- `length`: Length of the variables array.

int **thrd_set_global_env**(**const** char *name_p, **const** char *value_p)
Set the value of given environment variable. The pointers to given name and value are stored in the current global environment array.

**Return** zero(0) or negative error code.

**Parameters**

- `name_p`: Name of the environment variable to set.
- `value_p`: Value of the environment variable. Set to NULL to remove the variable.

**const** char ***thrd_get_global_env**(**const** char *name_p)
Get the value of given environment variable in the global environment array.

**Return** Value of given environment variable or NULL if it is not found.

**Parameters**

- `name_p`: Name of the environment variable to get.

int **thrd_init_env**(**struct** *thrd_environment_variable_t* *variables_p, int *length*)
Initialize the current threads' environment variables storage.

**Return** zero(0) or negative error code.

**Parameters**

- `variables_p`: Variables are to be used by this therad.
- `length`: Length of the variables array.

int **thrd_set_env**(**const** char *name_p, **const** char *value_p)
Set the value of given environment variable. The pointers to given name and value are stored in the current threads' environment array.

**Return** zero(0) or negative error code.

**Parameters**

- `name_p`: Name of the environment variable to set.
- `value_p`: Value of the environment variable. Set to NULL to remove the variable.

**const** char ***thrd_get_env**(**const** char **name_p*)
Get the value of given environment variable. If given variable is not found in the current threads' environment array, the global environment array is searched.

**Return** Value of given environment variable or NULL if it is not found.

**Parameters**

- `name_p`: Name of the environment variable to get.

int **thrd_suspend_isr**(**const struct** *time_t* **timeout_p*)
Suspend current thread with the system lock taken (see `sys_lock()`) and wait to be resumed or a timeout occurs (if given).

**Return** zero(0), -ETIMEDOUT on timeout or other negative error code.

**Parameters**

- `timeout_p`: Time to wait to be resumed before a timeout occurs and the function returns.

int **thrd_resume_isr**(**struct** *thrd_t* **thrd_p*, int *err*)
Resume given thread from isr or with the system lock taken (see `sys_lock()`). If resumed thread is not yet suspended it will not be suspended on next suspend call to `thrd_suspend()` or `thrd_suspend_isr()`.

**Return** zero(0) or negative error code.

**Parameters**

- `thrd_p`: Thread id to resume.
- `err`: Error code to be returned by `thrd_suspend()` or `thrd_suspend_isr()`.

int **thrd_yield_isr**(void)
Yield current thread from isr (preemptive scheduler only) or with the system lock taken.

**Return** zero(0) or negative error code.

void ***thrd_stack_alloc**(size_t *size*)
Allocate a thread stack of given size.

**Return** The pointer to allocated thread stack, or NULL on error.

int **thrd_stack_free**(void **stack_p*)
Free given thread stack.

**Return** zero(0) or negative error code.

**const** void ***thrd_get_bottom_of_stack**(**struct** *thrd_t* **thrd_p*)
Get the pointer to given threads' bottom of stack.

**Return** The pointer to given threds' bottom of stack, or NULL on error.

**const** void ***thrd_get_top_of_stack**(**struct** *thrd_t* **thrd_p*)
Get the pointer to given threads' top of stack.

**Return** The pointer to given threds' top of stack, or NULL on error.

int **thrd_prio_list_init**(**struct** *thrd_prio_list_t* *self_p)
Initialize given prio list.

void **thrd_prio_list_push_isr**(**struct** *thrd_prio_list_t* *self_p, **struct** *thrd_prio_list_elem_t* *elem_p)
Push given element on given priority list. The priority list is a linked list with the highest priority thread first. The pushed element is added *after* any already pushed elements with the same thread priority.

**Return** void.

**Parameters**

- `self_p`: Priority list to push on.
- `elem_p`: Element to push.

**struct** *thrd_prio_list_elem_t* ***thrd_prio_list_pop_isr**(**struct** *thrd_prio_list_t* *self_p)
Pop the highest priority element from given priority list.

**Return** Poped element or NULL if the list was empty.

**Parameters**

- `self_p`: Priority list to pop from.

int **thrd_prio_list_remove_isr**(**struct** *thrd_prio_list_t* *self_p, **struct** *thrd_prio_list_elem_t* *elem_p)
Remove given element from given priority list.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Priority list to remove given element from.
- `elem_p`: Element to remove.

**struct thrd_environment_variable_t**
*#include <thrd.h>* A thread environment variable.

### Public Members

**const** char ***name_p**

**const** char ***value_p**

**struct thrd_environment_t**

### Public Members

**struct** *thrd_environment_variable_t* ***variables_p**

size_t **number_of_variables**

size_t **max_number_of_variables**

**struct thrd_t**

### Public Members

**struct** *thrd_prio_list_elem_t* **elem**

**struct thrd_t::@91 thrd_t::scheduler**

**struct** thrd_port_t **port**

int8_t **prio**

int8_t **state**

int **err**

uint8_t **log_mask**

**struct** *timer_t* ***timer_p**

**const** char ***name_p**

**struct** *thrd_t* ***next_p**

**struct thrd_t::@92 thrd_t::statistics**

size_t **stack_size**

## `time` — System time

This module implements wall clock time, date and low overhead microsecond timing functions.

The `time_micros*()` and `time_busy_wait_us()` functions are intended for bit banging drivers, requiring precise microsecond timing with very low overhead. The internal microsecond counter wraps around quite frequently, and it's recommended to only measure very short time periods. The maximum time that can be measured is port specific, and can be read at runtime with `time_micros_maximum()`.

---

Source code: src/kernel/time.h, src/kernel/time.c

Test code: tst/kernel/time/main.c

Test coverage: src/kernel/time.c

---

### Enums

**enum time_compare_t**

A comparsion result.

*Values:*

**time_compare_less_than_t** = 0

**time_compare_equal_t** = 1

**time_compare_greater_than_t** = 2

### Functions

int **time_get**(**struct** *time_t* *now_p)
Get current time in seconds and nanoseconds. The resolution of the time is implementation specific and may vary a lot between different architectures.

**Return** zero(0) or negative error code.

**Parameters**

- `now_p`: Read current time.

int **time_set**(**struct** *time_t* *new_p)
Set current time in seconds and nanoseconds.

**Return** zero(0) or negative error code.

**Parameters**

- `new_p`: New current time.

int **time_add**(**struct** *time_t* *res_p, **struct** *time_t* *left_p, **struct** *time_t* *right_p)
Add given times.

**Return** zero(0) or negative error code.

**Parameters**

- `res_p`: The result of the adding `left_p` to `right_p`.
- `left_p`: First operand.
- `right_p`: Second operand.

int **time_subtract**(**struct** *time_t* *res_p, **struct** *time_t* *left_p, **struct** *time_t* *right_p)
Subtract given times.

**Return** zero(0) or negative error code.

**Parameters**

- `res_p`: The result of the subtrancting `left_p` from `right_p`.
- `left_p`: The operand to subtract from.
- `right_p`: The operand to subtract.

**enum** *time_compare_t* **time_compare**(**struct** *time_t* *left_p, **struct** *time_t* *right_p)
Compare given times and return their relationship as `less than`, `equal`, or `greater than`.

**Return** The result of the comparsion.

**Parameters**

- `left_p`: First time to compare.
- `right_p`: Second time to compare.

int **time_unix_time_to_date**(**struct** *date_t* *date_p, **struct** *time_t* *time_p)
Convert given unix time to a date.

**Return** zero(0) or negative error code or negative error code.

**Parameters**

- `date_p`: Converted time.
- `time_p`: Unix time to convert.

void **`time_busy_wait_us`**(int *microseconds*)

Busy wait for given number of microseconds.

This function may be called from interrupt context and with the system lock taken.

NOTE: The maximum allowed time to sleep is target specific.

**Return** void

**Parameters**

- `useconds`: Microseconds to busy wait.

int **`time_micros`**(void)

Get current time in microseconds. Use `time_micros_resolution()` and `time_micros_maximum()` to get its properties, and `time_micros_elapsed()` to calculate the elapsed time between two times.

This function may be called from interrupt context and with the system lock taken.

**Return** Current time in microseconds.

int **`time_micros_elapsed`**(int *start*, int *stop*)

Calculate the elapsed time from start to stop. The caller must ensure that the micro timer has not wrapped more than once for this calculation to work.

This function may be called from interrupt context and with the system lock taken.

**Return** The elapsed time from start to stop.

int **`time_micros_resolution`**(void)

Get micros resolution in microseconds, rounded up.

This function may be called from interrupt context and with the system lock taken.

**Return** Resolution in microseconds or negative error code.

int **`time_micros_maximum`**(void)

Get micros maximum value plus one, often the system tick period.

This function may be called from interrupt context and with the system lock taken.

**Return** Maximum value plus one in microseconds or negative error code. Returns -ENOSYS if the the micro functionality is unimplemented on this board.

**`struct time_t`**

*#include <time.h>* A time in seconds and nanoseconds. `seconds` and `nanoseconds` shall be added to get the time.

#### Public Members

int32_t **seconds**
Number of seconds.

int32_t **nanoseconds**
Number of nanoseconds.

**struct date_t**
*#include <time.h>* A date in year, month, date, day, hour, minute and seconds.

#### Public Members

int **second**
Second [0..59].

int **minute**
Minute [0..59].

int **hour**
Hour [0..23].

int **day**
Weekday [1..7], where 1 is Monday and 7 is Sunday.

int **date**
Day in month [1..31]

int **month**
Month [1..12] where 1 is January and 12 is December.

int **year**
Year [1970..].

### `timer` — Timers

Timers are started with a timeout, and when the time is up the timer expires and the timer callback function is called from interrupt context.

The timeout resolution is the system tick period. Timeouts are always rounded up to the closest system tick. That is, a timer can never expire early, but may expire slightly late.

Some ports support high resolution single shot timers, which often have higher resolution than the system tick.

---

Source code: src/kernel/timer.h, src/kernel/timer.c

Test code: tst/kernel/timer/main.c

Test coverage: src/kernel/timer.c

---

### Defines

**TIMER_PERIODIC**
A timer is "single shot" per default. Initialize a timer with this flag set in the `flags` argument to configure it as periodic.

A periodic timer will call the function callback periodically. This continues until the timer is stopped.

### Typedefs

**typedef** void (***timer_callback_t**)(void *arg_p)
Timer callback prototype.

### Functions

int **timer_module_init**(void)
Initialize the timer module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **timer_init**(**struct** *timer_t *self_p*, **const struct** *time_t *timeout_p*, *timer_callback_t callback*, void *arg_p, int *flags*)
Initialize given timer object with given timeout and expiry callback. The timer resolution directly depends on the system tick frequency and is rounded up to the closest possible value. This applies to both single shot and periodic timers. Some ports support high resolution timers, which often have higher resolution than the system tick.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Timer object to initialize with given parameters.
- `timeout_p`: The timer timeout value.
- `callback`: Functaion called when the timer expires. Called from interrupt context.
- `arg_p`: Function callback argument. Passed to the callback when the timer expires.
- `flags`: Set TIMER_PERIODIC for periodic timer.

int **timer_start**(**struct** *timer_t *self_p*)
Start given initialized timer object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Timer object to start.

int **timer_start_isr**(**struct** *timer_t *self_p*)
See `timer_start()` for a description.

This function may only be called from an isr or with the system lock taken (see `sys_lock()`).

int **timer_stop**(**struct** *timer_t* *self_p)

Stop given timer object. This has no effect on a timer that already expired or was never started. A stopped timer may be restarted with the initial timeout by calling `timer_start()`.

**Return** true(1) if the timer was stopped, false(0) if the timer already expired or was never started, and otherwise negative error code.

**Parameters**

- `self_p`: Timer object to stop.

int **timer_stop_isr**(**struct** *timer_t* *self_p)

See `timer_stop()` for description.

This function may only be called from an isr or with the system lock taken (see `sys_lock()`).

**struct timer_t**

### Public Members

**struct** *timer_t* ***next_p**

uint32_t **delta**

uint32_t **timeout**

int **flags**

*timer_callback_t* **callback**

void ***arg_p**

## types — Common types

A bunch of types and difinitions that didn't fit anywhere else.

---

Source code: src/kernel/types.h

---

### Defines

**UNUSED**(v)

Ignore unused function argument.

An example of a function that does not use it's first argument `a`:

```
int foo(int a, int b)
{
    UNUSED(a);

    return (b);
}
```

**STRINGIFY**(x)

Create a string of an identifier using the pre-processor.

**STRINGIFY2**(x)
Used internally by `STRINGIFY()`.

**TOKENPASTE**(x, y)
Concatenate two tokens.

**TOKENPASTE2**(x, y)
Used internally by `TOKENPASTE()`.

**UNIQUE**(x)
Create a unique token.

**PRINT_FILE_LINE**()
Debug print of file and line.

**STD_PRINTF_DEBUG**(...)

**membersof**(a)
Get the number of elements in an array.

As an example, the code below outputs `number of members in foo = 10`.

```
int foo[10];

std_printf(FSTR("number of members in foo = %d\\r\\n"),
           membersof(foo));
```

**indexof**(e_p, a)
Get the index of given element in an array.

**container_of**(ptr, type, member)

**DIV_CEIL**(n, d)
Integer division that rounds the result up.

**DIV_ROUND**(n, d)
Integer division that rounds the result to the closest integer.

**MIN**(a, b)
Get the minimum value of the two.

**MAX**(a, b)
Get the maximum value of the two.

**BIT**(pos)

**BITFIELD_SET**(name, value)

**BITFIELD_GET**(name, value)

**OSTR**(string)

**CSTR**(string)

### Typedefs

**typedef** uint8_t **u8_t**

**typedef** int8_t **s8_t**

**typedef** uint16_t **u16_t**

**typedef** int16_t **s16_t**

**typedef** uint32_t **u32_t**

**typedef** int32_t **s32_t**

#### Functions

**static** size_t **iov_uintptr_size**(**struct** *iov_uintptr_t* *iov_p, size_t *length*)

**struct thrd_prio_list_elem_t**

##### Public Members

**struct** *thrd_prio_list_elem_t* ***next_p**

**struct** *thrd_t* ***thrd_p**

**struct thrd_prio_list_t**

##### Public Members

**struct** *thrd_prio_list_elem_t* ***head_p**

**struct iov_t**

*#include* <*types.h*> Input-output vector.

##### Public Members

void ***buf_p**

size_t **size**

**struct iov_uintptr_t**

*#include* <*types.h*> Input-output vector with address.

##### Public Members

uintptr_t **address**

size_t **size**

### 1.6.2 sync

Thread synchronization refers to the idea that multiple threads are to join up or handshake at a certain point, in order to reach an agreement or commit to a certain sequence of action.

The sync package on Github.

#### bus — Message bus

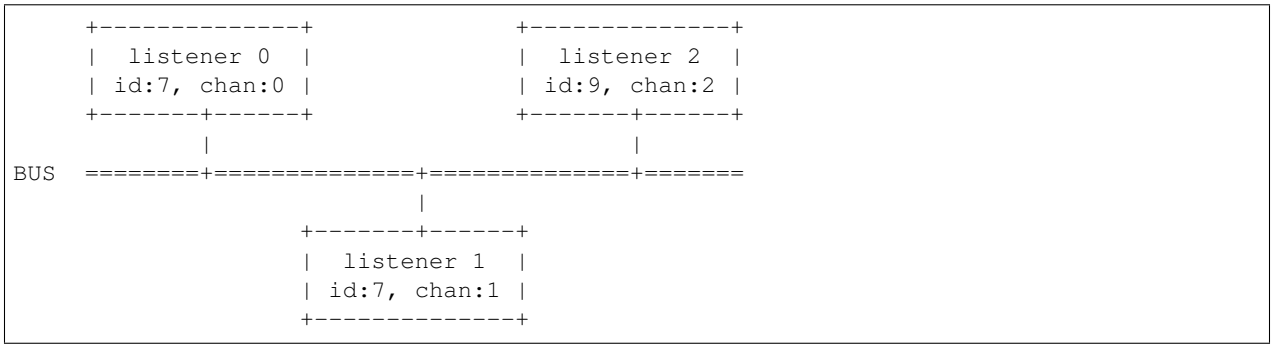
A message bus provides a software-bus abstraction that gathers all the communications between a group of threads over a single shared virtual channel. Messages are transferred on the bus from a sender to one or more attached listeners. The concept is analogous to the bus concept found in computer hardware architecture.

### Example

In this example there is a bus with three listeners attached; listerner *0*, *1* and *2*. Listener *0* and *1* are attached to the bus listening for message id *7*, and listener *2* for message id *9*.

Any thread can write a message to the bus by calling `bus_write()`. If a message with id *7* is written to the bus, both listerner *0* and *1* will receive the message. Listener *2* will receive messages with id *9*.

Messages are read from the listener channel by the thread that owns the listener.




---

Source code: src/sync/bus.h, src/sync/bus.c

Test code: tst/sync/bus/main.c

Test coverage: src/sync/bus.c

---

### Functions

int **bus_module_init**(void)

Initialize the bus module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code

int **bus_init**(**struct** *bus_t* **self_p*)

Initialize given bus.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Bus to initialize.

int **bus_listener_init**(**struct** *bus_listener_t* **self_p*, int *id*, void **chan_p*)

Initialize given listener to receive messages with given id, after the listener is attached to the bus. A listener can only receive messages of a single id, though, the same channel may be used in multiple listeners with different ids (if the channel supports it).

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Listener to initialize.
- `id`: Message id to receive.
- `chan_p`: Channel to receive messages on.

int **bus_attach**(**struct** *bus_t *self_p*, **struct** *bus_listener_t *listener_p*)

Attach given listener to given bus. Messages written to the bus will be written to all listeners initialized with the written message id.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Bus to attach the listener to.
- `listener_p`: Listener to attach to the bus.

int **bus_detach**(**struct** *bus_t *self_p*, **struct** *bus_listener_t *listener_p*)

Detatch given listener from given bus. A detached listener will not receive any messages from the bus.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Bus to detach listener from.
- `listener_p`: Listener to detach from the bus.

int **bus_write**(**struct** *bus_t *self_p*, int *id*, **const** void *buf_p*, size_t *size*)

Write given message to given bus. All attached listeners to given bus will receive the message.

**Return** Number of listeners that received the message, or negative error code.

**Parameters**

- `self_p`: Bus to write the message to.
- `id`: Message identity.
- `buf_p`: Buffer to write to the bus. All listeners with given message id will receive this data.
- `size`: Number of bytes to write.

**struct bus_t**

### Public Members

**struct** *rwlock_t* **rwlock**

**struct** *binary_tree_t* **listeners**

**struct bus_listener_t**

### Public Members

**struct** *binary_tree_node_t* **base**

int **id**

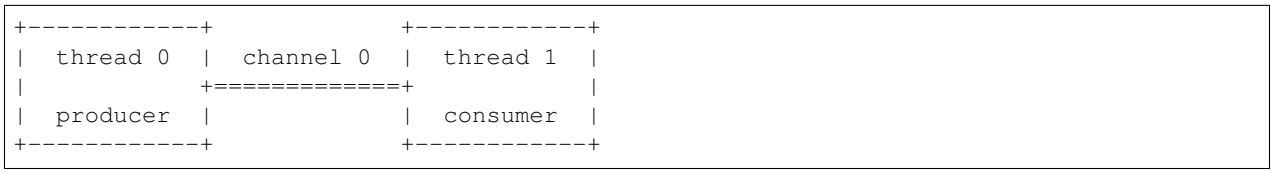
void ***chan_p**

**struct** *bus_listener_t* ***next_p**

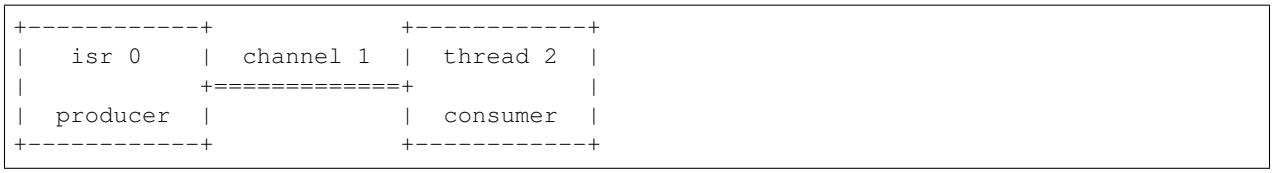
### `chan` — Abstract channel communication

Threads often communicate over channels. The producer thread or isr writes data to a channel and the consumer reads it. The may be multiple producers writing to a single channel, but only one consumer is allowed.

In the first example, `thread 0` and `thread 1` communicates over a channel. `thread 0` writes data to the channel and `thread 1` read the written data.




In the socond example, `isr 0` and `thread 2` communicates over a channel. `isr 0` writes data to the channel and `thread 2` read the written data.




---

Source code: src/sync/chan.h, src/sync/chan.c

Test coverage: src/sync/chan.c

---

#### Defines

**CHAN_CONTROL_LOG_BEGIN**
Beginning of a log entry.

**CHAN_CONTROL_LOG_END**
End of a log entry.

**CHAN_CONTROL_PRINTF_BEGIN**
Beginning of printf output.

**CHAN_CONTROL_PRINTF_END**
End of printf output.

**CHAN_CONTROL_NON_BLOCKING_READ**
Non-blocking read operation. A channel should return -EAGAIN when a read would block the channel.

**CHAN_CONTROL_BLOCKING_READ**
Blocking read operation.

### Typedefs

**typedef** ssize_t (***chan_read_fn_t**)(void *self_p, void *buf_p, size_t size)
Channel read function callback type.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: Channel to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

**typedef** ssize_t (***chan_write_fn_t**)(void *self_p, **const** void *buf_p, size_t size)
Channel write function callback type.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Channel to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

**typedef** int (***chan_control_fn_t**)(void *self_p, int operation)
Channel control function callback type.

**Return** Operation specific.

**Parameters**

- `self_p`: Channel to control.
- `operation`: Control operation.

**typedef** int (***chan_write_filter_fn_t**)(void *self_p, **const** void *buf_p, size_t size)
Channel write filter function callback type.

**Return** true(1) if the buffer shall be written to the channel, otherwise false(0).

**Parameters**

- `self_p`: Channel to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes in buffer.

**typedef** size_t (***chan_size_fn_t**)(void *self_p)
Channel size function callback type.

**Return** Number of bytes available.

**Parameters**

- `self_p`: Channel to get the size of.

### Functions

int **chan_module_init**(void)
Initialize the channel module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **chan_init**(**struct** *chan_t *self_p*, *chan_read_fn_t read*, *chan_write_fn_t write*, *chan_size_fn_t size*)
Initialize given channel with given callbacks. A channel must be initialized before it can be used.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Channel to initialize.
- `read`: Read function callback. This function must implement the channel read functionality, and will be called when the user reads data from the channel.
- `write`: Write function callback. This function must implement the channel write functionality, and will be called when the user writes data to the channel.
- `size`: Size function callback. This function must return the size of the channel. It should return zero(0) if there is no data available in the channel, and otherwise a positive integer.

int **chan_set_write_cb**(**struct** *chan_t *self_p*, *chan_write_fn_t write_cb*)
Set the write function callback.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `write_cb`: Write function to set.

int **chan_set_write_isr_cb**(**struct** *chan_t *self_p*, *chan_write_fn_t write_isr_cb*)
Set the write isr function callback.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `write_isr_cb`: Write isr function to set.

int **chan_set_write_filter_cb**(**struct** *chan_t *self_p*, *chan_write_filter_fn_t write_filter_cb*)
Set the write filter callback function. The write filter function is called when data is written to the channel, and its return value determines if the data shall be written to the underlying channel implementation, or discarded.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `write_filter_cb`: filter Write filter function to set.

int **chan_set_write_filter_isr_cb**(**struct** *chan_t* **self_p*, *chan_write_filter_fn_t write_filter_isr_cb*)

Set the write isr filter callback function. The write filter function is called when data is written to the channel, and its return value determines is the data shall be written to the underlying channel implementation, or discarded.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `write_filter_isr_cb`: filter Write filter function to set.

int **chan_set_control_cb**(**struct** *chan_t* **self_p*, *chan_control_fn_t control_cb*)

Set control function callback.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `control`: Control function to set.

ssize_t **chan_read**(void **self_p*, void **buf_p*, size_t *size*)

Read data from given channel. The behaviour of this function depends on the channel implementation. Often, the calling thread will be blocked until all data has been read or an error occurs.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: Channel to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **chan_write**(void **self_p*, **const** void **buf_p*, size_t *size*)

Write data to given channel. The behaviour of this function depends on the channel implementation. Some channel implementations blocks until the receiver has read the data, and some returns immediately.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Channel to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

int **chan_getc**(void **self_p*)

Read a character from given channel. The behaviour of this function depends on the channel implementation. Often, the calling thread will be blocked until the character has been read or an error occurs.

**Return** The read character as an unsigned char casted to an int, or negative error code.

**Parameters**

- `self_p`: Channel to read from.

int **chan_putc**(void *self_p, int character)

Write given character to given channel. The behaviour of this function depends on the channel implementation. Some channel implementations blocks until the receiver has read the character, and some returns immediately.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Channel to write to.
- `character`: Character to write as un unsigned char casted to an int.

size_t **chan_size**(void *self_p)

Get the number of bytes available to read from given channel.

**Return** Number of bytes available.

**Parameters**

- `self_p`: Channel to get the size of.

int **chan_control**(void *self_p, int operation)

Control given channel.

**Return** Operation specific.

**Parameters**

- `self_p`: Channel to control.
- `operation`: Operation to perform.

ssize_t **chan_write_isr**(void *self_p, **const** void *buf_p, size_t size)

Write data to given channel from interrupt context or with the system lock taken. The behaviour of this function depends on the channel implementation. Some channel implementations blocks until the receiver has read the data, and some returns immediately.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Channel to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

int **chan_is_polled_isr**(**struct** *chan_t* *self_p)

Check if a channel is polled. May only be called from isr or with the system lock taken (see `sys_lock()`).

**Return** true(1) or false(0).

**Parameters**

- `self_p`: Channel to check.

int **chan_list_init**(**struct** *chan_list_t* *self_p, **struct** *chan_list_elem_t* *elements_p, size_t number_of_elements)

Initialize an empty list of channels. A list is used to wait for data on multiple channel at the same time. When there is data on at least one channel, the poll function returns and the application can read from the channel with data.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: List to initialize.
- `elements_p`: Array of elements to store added channels in.
- `number_of_elements`: Number of elements in the element array.

int **chan_list_destroy**(**struct** *chan_list_t* *self_p)
Destroy an initialized list of channels.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: List to destroy.

int **chan_list_add**(**struct** *chan_list_t* *self_p, void *chan_p)
Add given channel to list of channels.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: List of channels.
- `chan_p`: Channel to add.

int **chan_list_remove**(**struct** *chan_list_t* *self_p, void *chan_p)
Remove given channel from list of channels.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: List of channels.
- `chan_p`: Channel to remove.

void ***chan_list_poll**(**struct** *chan_list_t* *self_p, **const struct** *time_t* *timeout_p)
Poll given list of channels for events. Blocks until at least one of the channels in the list has data ready to be read or an timeout occurs.

**Return** Channel with data or NULL on timeout.

**Parameters**

- `self_p`: List of channels to poll.
- `timeout_p`: Time to wait for data on any channel before a timeout occurs. Set to NULL to wait forever.

void ***chan_poll**(void *chan_p, **const struct** *time_t* *timeout_p)
Poll given channel for events. Blocks until the channel has data ready to be read or an timeout occurs.

**Return** The channel or NULL on timeout.

**Parameters**

- `chan_p`: Channel to poll.

- `timeout_p`: Time to wait for data on the channel before a timeout occurs. Set to NULL to wait forever.

void *__chan_null__(void)
Get a reference to the null channel. This channel will ignore all written data but return that it was successfully written.

**Return** The null channel.

ssize_t __chan_read_null__(void *_self_p_, void *_buf_p_, size_t _size_)
Null channel read function callback. Pass to `chan_init()` if no read function callback is needed for the channel.

**Return** Always returns -1.

**Parameters**

- `self_p`: Channel to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t __chan_write_null__(void *_self_p_, __const__ void *_buf_p_, size_t _size_)
Null channel write function callback. Pass to `chan_init()` if no write function callback is needed for the channel.

**Return** Always returns `size`.

**Parameters**

- `self_p`: Channel to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

size_t __chan_size_null__(void *_self_p_)
Null channel size function callback. Pass to `chan_init()` if no size function callback is needed for the channel.

**Return** Always returns zero(0).

**Parameters**

- `self_p`: Channel to get the size of.

int __chan_control_null__(void *_self_p_, int _operation_)
Null channel control function callback. Will silently ignore the control request.

**Return** Always returns zero(0).

**Parameters**

- `self_p`: Channel to control.
- `operation`: Operation to perform.

__struct chan_list_elem_t__

#### Public Members

**struct** *chan_t* ***chan_p**

**struct chan_list_t**

#### Public Members

**struct** *chan_list_elem_t* ***elements_p**

size_t **number_of_elements**

size_t **len**

**struct chan_t**

*#include* <*chan.h*> Channel datastructure.

#### Public Members

*chan_read_fn_t* **read**

*chan_write_fn_t* **write**

*chan_size_fn_t* **size**

*chan_control_fn_t* **control**

*chan_write_filter_fn_t* **write_filter_cb**

*chan_write_fn_t* **write_isr**

*chan_write_filter_fn_t* **write_filter_isr_cb**

**struct** *thrd_t* ***reader_p**

**struct** *chan_list_t* ***list_p**

### cond — Condition variable

A condition variable is a synchronization primitive used to unblock zero, one or all thread(s) waiting for the condition variable.

### Example usage

This is a small example of signalling a waiting thread.

```
struct cond_t cond;
struct mutex_t mutex;
int resource = 0;

/* Initialize the condition variable and a mutex. */
cond_init(&cond);
mutex_init(&mutex);

/* This thread waits for a signal from the signaller thread. */
void *waiter_main()
```

(continues on next page)

(continued from previous page)

```
{
    mutex_lock(&mutex);
    cond_wait(&cond, &mutex, NULL)
    mutex_unlock(&mutex);
}

/* This thread signals the waiter thread. */
void *signaller_main()
{
    /* The mutex is optional when signalling a condition. */
    mutex_lock(&mutex);
    cond_signal(&cond)
    mutex_unlock(&mutex);
}
```

---

Source code: src/sync/cond.h, src/sync/cond.c

Test code: tst/sync/cond/main.c

Test coverage: src/sync/cond.c

---

### Functions

int **cond_module_init**(void)

Initialize the condition variable module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code

int **cond_init**(**struct** *cond_t *self_p*)

Initialize given condition variable object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Condition variable to initialize.

int **cond_wait**(**struct** *cond_t *self_p*, **struct** *mutex_t *mutex_p*, **struct** *time_t *timeout_p*)

Wait until given condition variable is unblocked or an timeout occurs. Given mutex must be locked when this function is called, and it is still locked when this function returns.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Condition variable to wait for.
- `mutex_p`: Mutex.

int **cond_signal**(**struct** *cond_t *self_p*)

Unblock one thread waiting on given condition variable. This function is a no-op if no threads are waiting on given condition variable.

**Return** One(1) if a thread was unblocked, zero(0) if no thread was unblocked, or negative error code.

**Parameters**

- `self_p`: Condition variable.

int **cond_broadcast**(**struct** *cond_t *self_p*)

Unblock all threads waiting on given condition variable.

**Return** Number of unblocked threads or negative error code.

**Parameters**

- `self_p`: Condition variable.

**struct cond_t**

#### Public Members

**struct** *thrd_prio_list_t* **waiters**

Wait list.

### event — Event channel

An event channel consists of a 32 bits bitmap, where each bit corresponds to an event state. If the bit is set, the event is active. Since an event only has two states, active and inactive, signalling the same event multiple times will just result in the event to be active. There is no internal counter of how "active" an event is, it's simply active or inactive.

### Example usage

This is a small example of writing an event from an interrupt handler to a thread.

```
struct event_t event;

/* The interrupt handler. */
ISR(foo)
{
    uint32_t mask;

    mask = 0x1;
    event_write_isr(&event, &mask, sizeof(mask));
}

/* The thread. */
void bar(void *arg_p)
{
    uint32_t mask;

    /* Must be called before any read from or write to the event
       channel. */
```

(continues on next page)

(continued from previous page)

```
    event_init(&event);

    mask = 0x1;
    event_read(&event, &mask, sizeof(mask))

    /* Do something with the event. */
}
```

---

Source code: src/sync/event.h, src/sync/event.c

Test code: tst/sync/event/main.c

Test coverage: src/sync/event.c

---

## Functions

int **event_init**(**struct** *event_t* **self_p*)
Initialize given event channel.

**Return** zero(0) or negative error code

**Parameters**

- `self_p`: Event channel to initialize.

ssize_t **event_read**(**struct** *event_t* **self_p*, void **buf_p*, size_t *size*)
Wait for one or more events to occur in given event mask. This function blocks until at least one of the events in the event mask has been set. When the function returns, given event mask has been overwritten with the events that actually occured. Read events are automatically cleared.

**Return** sizeof(mask) or negative error code.

**Parameters**

- `self_p`: Event channel object.
- `buf_p`: The mask of events to wait for. When the function returns the mask contains the events that have occured.
- `size`: Size to read (always sizeof(mask)).

ssize_t **event_try_read**(**struct** *event_t* **self_p*, void **buf_p*, size_t *size*)
Try to read one or more events in given event mask. This function returns immediately, even if no event has occured. When the function returns, given mask has been overwritten with the events that actually occured. Read events are automatically cleared.

**Return** sizeof(mask) on success, -EAGAIN if no event was read, and otherwise other negative error code.

**Parameters**

- `self_p`: Event channel object.
- `buf_p`: The mask of events to wait for. When the function returns the mask contains the events that have occured.

- `size`: Size to read (always sizeof(mask)).

ssize_t **event_write**(**struct** *event_t* *self_p, **const** void *buf_p, size_t *size*)
Write given event(s) to given event channel.

**Return** sizeof(mask) or negative error code.

**Parameters**

- `self_p`: Event channel object.
- `buf_p`: The mask of events to write.
- `size`: Must always be sizeof(mask).

ssize_t **event_write_isr**(**struct** *event_t* *self_p, **const** void *buf_p, size_t *size*)
Write given events to the event channel from isr or with the system lock taken (see `sys_lock()`).

**Return** sizeof(mask) or negative error code.

**Parameters**

- `self_p`: Event channel object.
- `buf_p`: The mask of events to write.
- `size`: Must always be sizeof(mask).

ssize_t **event_size**(**struct** *event_t* *self_p)
Checks if there are events active on the event channel.

**Return** one(1) is at least one event has occured, otherwise zero(0).

**Parameters**

- `self_p`: Event channel object.

int **event_clear**(**struct** *event_t* *self_p, uint32_t *mask*)
Clear given events on the event channel.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Event channel object.
- `mask`: The mask of events to clear.

**struct event_t**
*#include* <*event.h*> Event channel.

### Public Members

**struct** *chan_t* **base**

uint32_t **mask**

uint32_t **reader_mask**

### mutex — Mutual exclusion

A mutex is a synchronization primitive used to protect a shared resource.

### Example usage

This is a small example of protecting a shared resource with a mutex.

```
struct mutex_t mutex;
int resource = 0;

/* Initialize the mutex. */
mutex_init(&mutex);

/* Increment the shared resource by one. */
mutex_lock(&mutex);
resource++;
mutex_unlock(&mutex);
```

---

Source code: src/sync/mutex.h, src/sync/mutex.c

Test code: tst/sync/mutex/main.c

Test coverage: src/sync/mutex.c

---

### Functions

int **mutex_module_init**(void)
Initialize the mutex module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code

int **mutex_init**(**struct** *mutex_t *self_p*)
Initialize given mutex object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Mutex to initialize.

int **mutex_lock**(**struct** *mutex_t *self_p*)
Lock given mutex.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Mutex to lock.

int **mutex_unlock**(**struct** *mutex_t *self_p*)
Unlock given mutex.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Mutex to unlock.

int **mutex_lock_isr**(**struct** *mutex_t *self_p*)
Lock given mutex with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Mutex to lock.

int **mutex_unlock_isr**(**struct** *mutex_t *self_p*)
Unlock given mutex with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Mutex to unlock.

**struct mutex_t**

#### Public Members

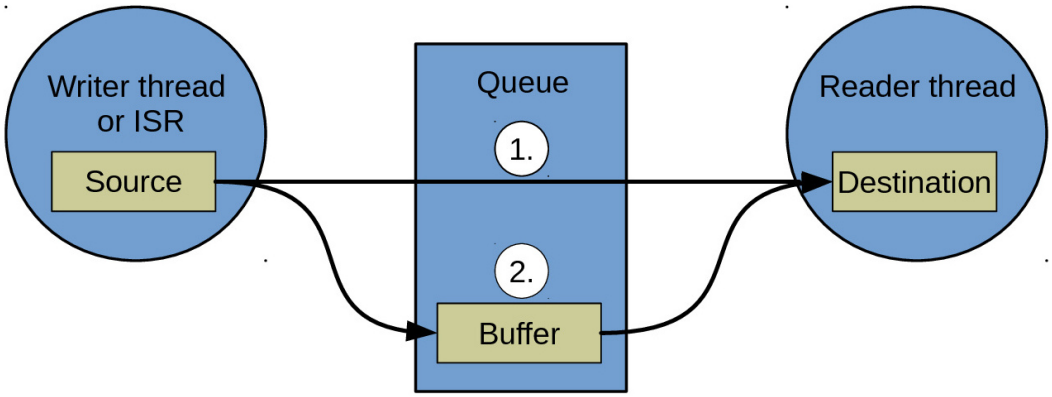
int8_t **is_locked**
Mutex lock state.

**struct** *thrd_prio_list_t* **waiters**
Wait list.

### queue — Queue channel

The most common channel is the queue. It can be either synchronous or semi-asynchronous. In the synchronous version the writing thread will block until all written data has been read by the reader. In the semi-asynchronous version the writer writes to a buffer within the queue, and only blocks all data does not fit in the buffer. The buffer size is selected by the application when initializing the queue.

The diagram below shows how two threads communicates using a queue. The writer thread writes from its source buffer to the queue. The reader thread reads from the queue to its destination buffer.




The data is either copied directly from the source to the destination buffer (1. in the figure), or via the internal queue buffer (2. in the figure).

1. The reader thread is waiting for data. The writer writes from its source buffer directly to the readers' destination buffer.
2. The reader thread is *not* waiting for data. The writer writes from its source buffer into the queue buffer. Later, the reader reads data from the queue buffer to its destination buffer.

### Example usage

This is a small example of writing a value from an interrupt handler to a thread.

```
struct queue_t queue;
uint8_t buf[8];

/* The interrupt handler. */
ISR(foo)
{
    uint8_t byte;

    byte = 1;
    queue_write_isr(&queue, &byte, sizeof(byte));
}

/* The thread. */
void bar(void *arg_p)
{
    uint8_t byte;

    /* Must be called before any read from or write to the
       queue. */
    queue_init(&queue, &buf[0], sizeof(buf));

    queue_read(&queue, &byte, sizeof(byte))

    /* Do something with the read byte. */
}
```

Source code: src/sync/queue.h, src/sync/queue.c

Test code: tst/sync/queue/main.c

Test coverage: src/sync/queue.c

Example code: examples/queue/main.c

---

### Defines

**QUEUE_FLAGS_NON_BLOCKING_READ**

**QUEUE_INIT_DECL**(_name, _buf, _size)

### Enums

**enum queue_state_t**

*Values:*

**QUEUE_STATE_INITIALIZED** = 0
Queue initialized state.

**QUEUE_STATE_RUNNING**
Queue running state.

**QUEUE_STATE_STOPPED**
Queue stopped state.

### Functions

int **queue_init**(**struct** *queue_t* *self_p, void *buf_p, size_t *size*)
Initialize given queue with given optional buffer.

**Return** zero(0) or negative error code

**Parameters**

- `self_p`: Queue to initialize.
- `buf_p`: Buffer for data storage. Give as NULL to disable buffering and only allow the writer to write directly into the reader's buffer, blocking the writer until all data has been written.
- `size`: Size of given buffer.

int **queue_start**(**struct** *queue_t* *self_p)
Start given queue. It is not required to start a queue unless it has been stopped.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Queue to start.

int **queue_stop**(**struct** *queue_t* *self_p)
Stop given queue. Any ongoing read and write operations will return with the currently read/written number of bytes. Any read and write operations on a stopped queue will return zero(0).

**Return** true(1) if a thread was resumed, false(0) if no thread was resumed, or negative error code.

**Parameters**

- `self_p`: Queue to stop.

int **queue_stop_isr**(**struct** *queue_t* *self_p)
Same as `queue_stop()` but from isr or with the system lock taken (see `sys_lock()`).

ssize_t **queue_read**(**struct** *queue_t* *self_p, void *buf_p, size_t size)
Read from given queue. Blocks until size bytes has been read.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Queue to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **queue_write**(**struct** *queue_t* *self_p, **const** void *buf_p, size_t size)
Write bytes to given queue. Blocks until size bytes has been written.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Queue to write to.
- `buf_p`: Buffer to write from.
- `size`: Number of bytes to write.

ssize_t **queue_write_isr**(**struct** *queue_t* *self_p, **const** void *buf_p, size_t size)
Write bytes to given queue from isr or with the system lock taken (see `sys_lock()`). May write less than size bytes.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Queue to write to.
- `buf_p`: Buffer to write from.
- `size`: Number of bytes to write.

ssize_t **queue_size**(**struct** *queue_t* *self_p)
Get the number of bytes currently stored in the queue. May return less bytes than number of bytes stored in the channel.

**Return** Number of bytes in given queue.

**Parameters**

- `self_p`: Queue.

ssize_t **queue_unused_size**(**struct** *queue_t* *self_p)
Get the number of unused bytes in the queue.

**Return** Number of unused bytes in given queue.

**Parameters**

- `self_p`: Queue.

ssize_t **queue_unused_size_isr**(**struct** *queue_t* **self_p*)
Get the number of unused bytes in the queue from isr or with the system lock taken (see `sys_lock()`).

**Return** Number of unused bytes in given queue.

**Parameters**

- `self_p`: Queue.

ssize_t **queue_ignore**(**struct** *queue_t* **self_p*, size_t *size*)
Ignore given number of bytes at the beginning of the queue by discarding them.

**Return** Number of bytes ignored or negative error code.

**Parameters**

- `self_p`: Queue.

**struct queue_t**

### Public Members

**struct** *chan_t* **base**

**struct** *thrd_prio_list_t* **writers**

**struct** queue_writer_elem_t ***writer_p**

char ***buf_p**

size_t **size**

size_t **left**

**struct queue_t::@125 queue_t::reader**

void ***buf_p**

**struct** *circular_buffer_t* **buffer**

*queue_state_t* **state**

int **flags**

## `rwlock` — Reader-writer lock

An RW lock allows concurrent access for read-only operations, while write operations require exclusive access. This means that multiple threads can read the data in parallel but an exclusive lock is needed for writing or modifying data. When a writer is writing the data, all other writers or readers will be blocked until the writer is finished writing. A common use might be to control access to a data structure in memory that cannot be updated atomically and is invalid (and should not be read by another thread) until the update is complete.

---

Source code: src/sync/rwlock.h, src/sync/rwlock.c

Test code: tst/sync/rwlock/main.c

Test coverage: src/sync/rwlock.c

---

### Functions

int **rwlock_module_init**(void)
Initialize the reader-writer lock module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code

int **rwlock_init**(**struct** *rwlock_t* **self_p*)
Initialize given reader-writer lock object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Reader-writer lock to initialize.

int **rwlock_reader_take**(**struct** *rwlock_t* **self_p*)
Take given reader-writer lock. Multiple threads can have the reader lock at the same time.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Reader-writer lock to take.

int **rwlock_reader_give**(**struct** *rwlock_t* **self_p*)
Give given reader-writer lock.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Reader-writer lock give.

int **rwlock_reader_give_isr**(**struct** *rwlock_t* **self_p*)
Give given reader-writer lock from isr or with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Reader-writer lock to give.

int **rwlock_writer_take**(**struct** *rwlock_t* **self_p*)
Take given reader-writer lock as a writer. Only one thread can have the lock at a time, including both readers and writers.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Reader-writer lock to take.

int **rwlock_writer_give**(**struct** *rwlock_t *self_p*)
Give given reader-writer lock.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Reader-writer lock to give.

int **rwlock_writer_give_isr**(**struct** *rwlock_t *self_p*)
Give given reader-writer lock from isr or with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Reader-writer lock to give.

**struct rwlock_t**

#### Public Members

int **number_of_readers**

int **number_of_writers**

**volatile struct** rwlock_elem_t ***readers_p**

**volatile struct** rwlock_elem_t ***writers_p**

### sem — Counting semaphores

The semaphore is a synchronization primitive used to protect a shared resource. A semaphore counts the number of resources taken, and suspends threads when the maximum number of resources are taken. When a resource becomes available, a suspended thread is resumed.

A semaphore initialized with *count_max* one(1) is called a binary semaphore. A binary semaphore can only be taken by one thread at a time and can be used to signal that an event has occured. That is, *sem_give()* may be called multiple times and the semaphore resource count will remain at zero(0) until *sem_take()* is called.

### Example usage

This is a small example of protecting a shared resource with a semaphore.

```
struct sem_t sem;
int resource = 0;

/* Initialize the semaphore. */
sem_init(&sem, 0, 1);

/* Increment the shared resource by one. */
sem_take(&sem, NULL);
resource++;
sem_give(&sem, 1);
```

Source code: src/sync/sem.h, src/sync/sem.c

Test code: tst/sync/sem/main.c

Test coverage: src/sync/sem.c

## Defines

**SEM_INIT_DECL**(name, _count, _count_max)
Compile-time declaration of a semaphore.

**Parameters**

- `name`: Semaphore to initialize.
- `count`: Initial count. Set the initial count to the same value as count_max to initialize the semaphore with all resources used.
- `count_max`: Maximum number of users holding the semaphore at the same time.

## Functions

int **sem_module_init**(void)
Initialize the semaphore module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code

int **sem_init**(**struct** *sem_t* *self_p*, int *count*, int *count_max*)
Initialize given semaphore object. Maximum count is the number of resources that can be taken at any given moment.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Semaphore to initialize.
- `count`: Initial taken resource count. Set the initial count to the same value as count_max to initialize the semaphore with all resources taken.
- `count_max`: Maximum number of resources that can be taken at any given moment.

int **sem_take**(**struct** *sem_t* *self_p*, **struct** *time_t* *timeout_p*)
Take given semaphore. If the semaphore count is zero the calling thread will be suspended until count is incremented by `sem_give()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Semaphore to take.
- `timeout_p`: Timeout.

int **sem_give**(**struct** *sem_t* *self_p*, int *count*)

Give given count to given semaphore. Any suspended thread waiting for this semaphore, in `sem_take()`, is resumed. This continues until the semaphore count becomes zero or there are no threads in the suspended list.

Giving a count greater than the currently taken count is allowed and results in all resources available. This is especially useful for binary semaphores where `sem_give()` if often called more often than `sem_take()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Semaphore to give count to.
- `count`: Count to give.

int **sem_give_isr**(**struct** *sem_t* *self_p*, int *count*)

Give given count to given semaphore from isr or with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Semaphore to give count to.
- `count`: Count to give.

**struct sem_t**

#### Public Members

int **count**

Number of used resources.

int **count_max**

Maximum number of resources.

**struct** *thrd_prio_list_t* **waiters**

Wait list.

### 1.6.3 drivers

The drivers package on Github.

Modules:

#### basic

Basic drivers.

#### adc — Analog to digital convertion

Source code: src/drivers/basic/adc.h, src/drivers/basic/adc.c

Test code: tst/drivers/hardware/basic/adc/main.c

### Defines

**ADC_REFERENCE_VCC**
Use VCC as reference for convertions.

### Functions

int **adc_module_init**(void)
Initialize the ADC driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **adc_init**(**struct** adc_driver_t *self_p, **struct** adc_device_t *dev_p, **struct** pin_device_t *pin_dev_p, int *reference*, long *sampling_rate*)
Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: ADC device to use.
- `pin_dev_p`: Pin device to use.
- `reference`: Voltage reference. Only `ADC_REFERENCE_VCC` is supported.
- `sampling_rate`: Sampling rate in Hz. The lowest allowed value is one and the highest value depends on the architechture. The sampling rate is not used in single sample convertions, ie. calls to `adc_async_convert()` and `adc_convert()` with length one; or calls to `adc_convert_isr()`.

int **adc_async_convert**(**struct** adc_driver_t *self_p, uint16_t *samples_p, size_t *length*)
Start an asynchronous convertion of analog signal to digital samples. Call `adc_async_wait()` to wait for the convertion to complete.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `samples_p`: Converted samples.
- `length`: Length of samples array.

int **adc_async_wait**(**struct** adc_driver_t *self_p)
Wait for an asynchronous convertion to complete.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **adc_convert**(**struct** adc_driver_t *self_p, uint16_t *samples_p, size_t length)

Start a synchronous convertion of an analog signal to digital samples. This is equivalent to `adc_async_convert()` + `adc_async_wait()`, but in a single function call.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `samples_p`: Converted samples.
- `length`: Length of samples array.

int **adc_convert_isr**(**struct** adc_driver_t *self_p, uint16_t *sample_p)

Start a synchronous convertion of analog signal to digital samples from isr or with the system lock taken. This function will poll the ADC hardware until the sample has been coverted.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `sample_p`: Converted sample.

int **adc_is_valid_device**(**struct** adc_device_t *dev_p)

Check if given ADC device is valid.

**Return** true(1) if the pin device is valid, otherwise false(0).

**Parameters**

- `dev_p`: ADC device to validate.

### Variables

**struct** adc_device_t **adc_device**[**ADC_DEVICE_MAX**]

## `analog_input_pin` — Analog input pin

Source code: src/drivers/basic/analog_input_pin.h, src/drivers/basic/analog_input_pin.c

Test code: tst/drivers/hardware/basic/analog_input_pin/main.c

---

### Functions

int **analog_input_pin_module_init**(void)

Initialize the analog input pin module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **analog_input_pin_init**(**struct** *analog_input_pin_t* *self_p, **struct** pin_device_t *dev_p)
Initialize given driver object with given device and mode.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Device to use.

int **analog_input_pin_read**(**struct** *analog_input_pin_t* *self_p)
Read the current value of given pin.

**Return** Analog pin value, otherwise negative error code.

**Parameters**

- `self_p`: Driver object.

int **analog_input_pin_read_isr**(**struct** *analog_input_pin_t* *self_p)
Read the current value of given pin from an isr or with the system lock taken.

**Return** Analog pin value, otherwise negative error code.

**Parameters**

- `self_p`: Driver object.

**struct analog_input_pin_t**

#### Public Members

**struct** adc_driver_t **adc**

### analog_output_pin — Analog output pin

Source code: src/drivers/basic/analog_output_pin.h, src/drivers/basic/analog_output_pin.c

Test code: tst/drivers/hardware/basic/analog_output_pin/main.c

---

#### Functions

int **analog_output_pin_module_init**(void)
Initialize the analog output pin module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **analog_output_pin_init**(**struct** *analog_output_pin_t* *self_p, **struct** pin_device_t *dev_p)
Initialize given driver object with given device and mode.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Device to use.

int **analog_output_pin_write**(**struct** *analog_output_pin_t* **self_p*, int *value*)
Write given value to the analog pin.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `value`: The value to write to the pin. A number in the range 0 to 1023, where 0 is lowest output and 1023 is highest output.

int **analog_output_pin_read**(**struct** *analog_output_pin_t* **self_p*)
Read the value that is currently written to given analog output pin.

**Return** Value in the range 0 to 1023, or negative error code.

**Parameters**

- `self_p`: Driver object.

**struct analog_output_pin_t**

#### Public Members

**struct** pwm_driver_t **pwm**

### `chipid` — Chip identity

Source code: src/drivers/basic/chipid.h, src/drivers/basic/chipid.c

Test code: tst/drivers/hardware/basic/chipid/main.c

---

#### Functions

int **chipid_read**(**struct** chipid_t **id_p*)
Read chipset identify from the hardware.

**Return** zero(0) or negative error code.

**Parameters**

- `id_p`: Read chip identity.

### dac — Digital to analog convertion

Source code: src/drivers/basic/dac.h, src/drivers/basic/dac.c

Test code: tst/drivers/hardware/basic/dac/main.c

---

## Functions

int **dac_module_init**(void)
Initialize DAC driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **dac_init**(**struct** dac_driver_t *self_p, **struct** dac_device_t *dev_p, **struct** pin_device_t *pin0_dev_p, **struct** pin_device_t *pin1_dev_p, int sampling_rate)
Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Device to use.
- `pin0_dev_p`: Pin used for mono or first stereo channel.
- `pin1_dev_p`: Second stereo pin.
- `sampling_rate`: Sampling rate in Hz.

int **dac_async_convert**(**struct** dac_driver_t *self_p, void *samples_p, size_t length)
Start an asynchronous convertion of samples to an analog signal.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `samples_p`: Samples to convert to an analog signal.
- `length`: Length of samples array.

int **dac_async_wait**(**struct** dac_driver_t *self_p)
Wait for ongoing asynchronous convertion to finish.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **dac_convert**(**struct** dac_driver_t *self_p, void *samples_p, size_t length)
Start synchronous convertion of samples to an analog signal.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `samples_p`: Converted samlpes.
- `length`: Length of samples array.

### Variables

**struct** dac_device_t **dac_device**[**DAC_DEVICE_MAX**]

### `exti` — External interrupts

Source code: src/drivers/basic/exti.h, src/drivers/basic/exti.c

Test code: tst/drivers/hardware/basic/exti/main.c

---

### Defines

**EXTI_TRIGGER_BOTH_EDGES**
Trigger an interrupt on both rising and falling edges.

**EXTI_TRIGGER_FALLING_EDGE**
Trigger an interrupt on falling edges.

**EXTI_TRIGGER_RISING_EDGE**
Trigger an interrupt on rising edges.

### Functions

int **exti_module_init**(void)
Initialize the external interrupt (EXTI) module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **exti_init**(**struct** exti_driver_t *self_p, **struct** exti_device_t *dev_p, int trigger, void (*on_interrupt)) void *arg_p
, void *arg_pInitialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Device to use.
- `trigger`: One of `EXTI_TRIGGER_BOTH_EDGES`, `EXTI_TRIGGER_FALLING_EDGE` or `EXTI_TRIGGER_RISING_EDGE`.

- `on_interrupt`: Function callback called when an interrupt occurs.
- `arg_p`: Function callback argument.

int **exti_start**(**struct** exti_driver_t *self_p)
Starts the EXTI device using given driver object. Enables interrupts for given external interrupt driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **exti_stop**(**struct** exti_driver_t *self_p)
Stops the EXTI device referenced by given driver object. Disables interrupts for given external interrupt driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **exti_clear**(**struct** exti_driver_t *self_p)
Clear the interrupt flag.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

### Variables

**struct** exti_device_t **exti_device**[**EXTI_DEVICE_MAX**]

## `pcint` — Pin change interrupts

This module adds support for external pin change interrupts, with triggers on rising and/or falling edges.

---

Source code: src/drivers/basic/pcint.h, src/drivers/basic/pcint.c

Test code: tst/drivers/hardware/basic/pcint/main.c

---

### Defines

**PCINT_TRIGGER_BOTH_EDGES**
Trigger an interrupt on both rising and falling edges.

**PCINT_TRIGGER_FALLING_EDGE**
Trigger an interrupt on falling edges.

**PCINT_TRIGGER_RISING_EDGE**
Trigger an interrupt on rising edges.

### Functions

int **pcint_module_init**(void)

Initialize the external change interrupt module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **pcint_init**(**struct** pcint_driver_t *self_p, **struct** pcint_device_t *dev_p, int trigger, void (*on_interrupt)) void *arg_p
, void *arg_pInitialize given change interrupt driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Device to use.
- `trigger`: One of `PCINT_TRIGGER_BOTH_EDGES`, `PCINT_TRIGGER_FALLING_EDGE` or `PCINT_TRIGGER_RISING_EDGE`.
- `on_interrupt`: Function callback called when an interrupt occurs.
- `arg_p`: Function callback argument.

int **pcint_start**(**struct** pcint_driver_t *self_p)

Starts the pin change interrupt device using given driver object.

Enables interrupts for given pin change interrupt driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **pcint_stop**(**struct** pcint_driver_t *self_p)

Stops the pin change interrupt device referenced by given driver object.

Disables interrupts for given pin change interrupt driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

### Variables

**struct** pcint_device_t **pcint_device**[**PCINT_DEVICE_MAX**]

### pin — Digital pins

### Debug file system commands

Three debug file system commands are available, all located in the directory drivers/pin/. These commands directly access the pin device registers, without using the pin driver object.

| Command | Description |
|---|---|
| set_mode <pin> <mode> | Set the mode of the pin <pin> to <mode>, where <mode> is one of output, output_open_drain, output_open_drain_pull_up, input, input_pull_up and input_pull_down. |
| read <pin> | Read current input value of the pin <pin>. high or low is printed. |
| write <pin> <value> | Write the value <value> to given output pin <pin>, where <value> is one of high and low. |

Example output from the shell:

```
$ drivers/basic/pin/set_mode d2 output
OK
$ drivers/basic/pin/write d2 high
OK
$ drivers/basic/pin/write d2 low
OK
$ drivers/basic/pin/set_mode d3 input
OK
$ drivers/basic/pin/read d3
low
OK
```

---

Source code: src/drivers/basic/pin.h, src/drivers/basic/pin.c

Test code: tst/drivers/hardware/basic/pin/main.c

---

### Defines

**PIN_OUTPUT**
Configure the pin as an output pin.

**PIN_OUTPUT_OPEN_DRAIN**
Configure the pin as an output open drain pin.

**PIN_OUTPUT_OPEN_DRAIN_PULL_UP**
Configure the pin as an output open drain pin with the internal pull-up resistor enabled.

**PIN_INPUT**
Configure the pin as an input pin.

**PIN_INPUT_PULL_UP**
Configure the pin as an input pin with the internal pull-up resistor enabled.

**PIN_INPUT_PULL_DOWN**
Configure the pin as an input pin with the internal pull-down resistor enabled.

### Functions

int **pin_module_init**(void)
Initialize the pin module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **pin_init**(**struct** pin_driver_t *self_p, **struct** pin_device_t *dev_p, int *mode*)
Initialize given driver object with given device and mode.

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Device to use.
- `mode`: Pin mode. One of the `PIN_OUTPUT*` and `PIN_INPUT*` defines.

int **pin_write**(**struct** pin_driver_t *self_p, int *value*)
Write given value to given pin.

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `value`: Non-zero for high and `0` for low output.

int **pin_read**(**struct** pin_driver_t *self_p)
Read the current value of given pin.

This function may be called from interrupt context and with the system lock taken.

**Return** `1` for high and `0` for low input, otherwise negative error code.

**Parameters**

- `self_p`: Driver object.

int **pin_toggle**(**struct** pin_driver_t *self_p)
Toggle the pin output value (high/low).

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **pin_set_mode**(**struct** pin_driver_t *self_p, int *mode*)
Set the pin mode of given pin.

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `mode`: Pin mode. One of the `PIN_OUTPUT*` and `PIN_INPUT*` defines.

**static** int **pin_device_set_mode**(**const struct** pin_device_t *dev_p, int *mode*)
Pin device mode to set.

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `dev_p`: Pin device.
- `mode`: Pin mode. One of the `PIN_OUTPUT*` and `PIN_INPUT*` defines.

**static** int **pin_device_read**(**const struct** pin_device_t *dev_p)
Read the value of given pin device.

This function may be called from interrupt context and with the system lock taken.

**Return** `1` for high and `0` for low input, otherwise negative error code.

**Parameters**

- `dev_p`: Pin device.

int **pin_device_write**(**const struct** pin_device_t *dev_p, int *value*)
Write given value to given pin device.

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `dev_p`: Pin device.
- `value`: Non-zero for high and `0` for low output.

**static** int **pin_device_write_high**(**const struct** pin_device_t *dev_p)
Write high to given pin device.

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `dev_p`: Pin device.

**static** int **pin_device_write_low**(**const struct** pin_device_t *_dev_p_)
Write low to given pin device.

This function may be called from interrupt context and with the system lock taken.

**Return** zero(0) or negative error code.

**Parameters**

- `dev_p`: Pin device.

int **pin_is_valid_device**(**struct** pin_device_t *_dev_p_)
Check if given pin device is valid.

This function may be called from interrupt context and with the system lock taken.

**Return** true(1) if the pin device is valid, otherwise false(0).

**Parameters**

- `dev_p`: Pin device to validate.

### Variables

**struct** pin_device_t **pin_device**[**PIN_DEVICE_MAX**]

## power — Power control

Source code: src/drivers/basic/power.h, src/drivers/basic/power.c

---

### Functions

int **power_module_init**(void)
Initialize the power module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **power_deep_sleep**(long *microseconds*)
Put board into deep sleep mode for requested time.

**Return** zero(0) or negative error code.

**Parameters**

- `microseconds`: Time to sleep.

## pwm — Pulse width modulation

Source code: src/drivers/basic/pwm.h, src/drivers/basic/pwm.c

---

### Functions

int **pwm_module_init**(void)
Initialize the pwm module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **pwm_init**(**struct** pwm_driver_t *self_p, **struct** pwm_device_t *dev_p, long *frequency*, long *duty_cycle*)
Initialize given PWM driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: PWM device to use.
- `frequency`: Frequency.
- `duty_cycle`: Duty cycle.

int **pwm_start**(**struct** pwm_driver_t *self_p)
Start given PWM driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to start.

int **pwm_stop**(**struct** pwm_driver_t *self_p)
Stop given PWM driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to stop.

int **pwm_set_frequency**(**struct** pwm_driver_t *self_p, long *value*)
Set the frequency of the PWM signal.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `value`: Frequency. Use `pwm_frequency()` to convert a frequency in Hertz to a value expected by this function.

long **pwm_get_frequency**(**struct** pwm_driver_t *self_p)
Get current frequency.

**Return** Current frequency.

**Parameters**

- `self_p`: Driver object.

int **pwm_set_duty_cycle**(**struct** pwm_driver_t *self_p, long *value*)
Set the duty cycle of the signal.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `value`: Duty cycle. Use `pwm_duty_cycle()` to convert a duty cycle percentage to a value expected by this function.

long **pwm_get_duty_cycle**(**struct** pwm_driver_t *self_p)
Get current duty cycle.

**Return** Current duty cycle.

**Parameters**

- `self_p`: Driver object.

long **pwm_frequency**(int *hertz*)
Convert a duty cycle percentage to a value for `pwm_set_frequency()`.

**Return** Frequency.

**Parameters**

- `hertz`: Frequency in Hertz.

int **pwm_frequency_as_hertz**(long *value*)
Convert a frequency value returned by `pwm_get_frequency()` to Hertz.

**Return** Frequency in Hertz.

**Parameters**

- `value`: Frequency.

long **pwm_duty_cycle**(int *percentage*)
Convert a duty cycle percentage to a value for `pwm_set_duty_cycle()`.

**Return** Duty cycle.

**Parameters**

- `percentage`: Duty cycle percentage.

int **pwm_duty_cycle_as_percent**(long *value*)
Convert a duty cycle value returned by `pwm_get_duty_cycle()` to a percentage.

**Return** Duty cycle percentage.

**Parameters**

- `value`: Duty cycle.

**struct** pwm_device_t ***pwm_pin_to_device**(**struct** pin_device_t **pin_p*)
Get the PWM device for given pin.

**Return** PWM device, or NULL on error.

**Parameters**

- `pin_p`: The pin device to get the PWM device for.

### Variables

**struct** pwm_device_t **pwm_device**[**PWM_DEVICE_MAX**]

## `pwm_soft` — Software pulse width modulation

This module implements software PWM on all digital pins. In general, software PWM outputs an inaccurate, low frequency signal. Keep that in mind designing your application.

If an accurate and/or high frequency PWM signal is required, a *hardware PWM* should be used instead.

Here is a short example of how to use this module. A software PWM driver is initialized for digital pin 3 (D3). A software PWM signal with duty cycle 10% is outputted on D3 after the calling *pwm_soft_start()*.

```
struct pwm_soft_driver_t pwm_soft;

pwm_soft_module_init(500);
pwm_soft_init(&pwm_soft, &pin_d3_dev, pwm_soft_duty_cycle(10));
pwm_soft_start(&pwm_soft);
```

Change the duty cycle to 85% by calling *pwm_soft_set_duty_cycle()*.

```
pwm_soft_set_duty_cycle(&pwm_soft, pwm_soft_duty_cycle(85));
```

Stop outputting the software PWM signal to D3 by calling *pwm_soft_stop()*.

```
pwm_soft_stop(&pwm_soft);
```

---

Source code: src/drivers/basic/pwm_soft.h, src/drivers/basic/pwm_soft.c

Test code: tst/drivers/hardware/basic/pwm_soft/main.c

---

### Functions

int **pwm_soft_module_init**(long *frequency*)
Initialize the software PWM module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

**Parameters**

- `frequency`: PWM module frequency in Hertz. All software PWM:s will run at this frequency. The frequency can later be changed by calling `pwm_soft_set_frequency()`.

int **pwm_soft_set_frequency**(long *value*)

Set the frequency. The frequency is the same for all software PWM:s. All software PWM:s must be stopped before calling this function, otherwize a negative error code will be returned.

**Return** zero(0) or negative error code.

**Parameters**

- `value`: Frequency to set in Hertz. All software PWM:s will run at this frequency.

long **pwm_soft_get_frequency**(void)

Get current frequency.

**Return** Current frequency in Hertz.

int **pwm_soft_init**(**struct** *pwm_soft_driver_t* **self_p*, **struct** pin_device_t **pin_dev_p*, long *duty_cycle*)

Initialize given software PWM driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `pin_dev_p`: Pin device to use.
- `duty_cycle`: Initial duty cycle.

int **pwm_soft_start**(**struct** *pwm_soft_driver_t* **self_p*)

Start outputting the PWM signal on the pin given to `pwm_soft_init()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to start.

int **pwm_soft_stop**(**struct** *pwm_soft_driver_t* **self_p*)

Stop outputting the PWM signal on the pin given to `pwm_soft_init()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to stop.

int **pwm_soft_set_duty_cycle**(**struct** *pwm_soft_driver_t* **self_p*, long *value*)

Set the duty cycle. Calls `pwm_soft_stop()` and `pwm_soft_start()` to restart outputting the PWM signal with the new duty cycle.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

- `value`: Duty cycle. Use `pwm_soft_duty_cycle()` to convert a duty cycle percentage to a value expected by this function.

unsigned int **pwm_soft_get_duty_cycle**(**struct** *pwm_soft_driver_t* **self_p*)

Get current duty cycle. Use `pwm_soft_duty_cycle_as_percent()` to convert a duty cycle to a percentage.

**Return** Current duty cycle.

**Parameters**

- `self_p`: Driver object.

long **pwm_soft_duty_cycle**(int *percentage*)

Convert a duty cycle percentage to a value for `pwm_soft_init()` and `pwm_soft_set_duty_cycle()`.

**Return** Duty cycle.

**Parameters**

- `percentage`: Duty cycle percentage.

int **pwm_soft_duty_cycle_as_percent**(long *value*)

Convert a duty cycle value for `pwm_soft_init()` and `pwm_soft_set_duty_cycle()` to a percentage.

**Return** Duty cycle percentage.

**Parameters**

- `value`: Duty cycle.

**struct pwm_soft_driver_t**

#### Public Members

**struct** pin_device_t ***pin_dev_p**

long **frequency**

long **duty_cycle**

unsigned int **delta**

**struct** *thrd_t* ***thrd_p**

**struct** *pwm_soft_driver_t* ***next_p**

### `random` — Random numbers.

Source code: src/drivers/basic/random.h, src/drivers/basic/random.c

Test code: tst/drivers/hardware/basic/random/main.c

### Functions

int **random_module_init**(void)
Initialize the random module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

uint32_t **random_read**(void)
Read a 32 bits random number from the hardware.

**Return** A 32 bits random number.

## watchdog — Hardware watchdog

Source code: src/drivers/basic/watchdog.h, src/drivers/basic/watchdog.c

---

### Typedefs

**typedef** void (***watchdog_isr_fn_t**)(void)
Watchdog interrupt function prototype.

### Functions

int **watchdog_module_init**(void)
Initialize the watchdog driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **watchdog_start_ms**(int *timeout*, *watchdog_isr_fn_t* *on_interrupt*)
Start the watchdog with given timeout. Use `watchdog_kick()` to periodically restart the timer.

**Return** zero(0) or negative error code.

**Parameters**

- `timeout`: Watchdog timeout in milliseconds. The timeout is rounded up to the closest timeout the hardware supports.
- `on_interrupt`: Function callback called when a watchdog interrupt occurs. Not all MCU:s supports this feature.

int **watchdog_stop**(void)
Stop the watchdog.

**Return** zero(0) or negative error code.

int **watchdog_kick**(void)

Kick the watchdog. Restarts the watchdog timer with its original timeout given to `watchdog_start_ms()`. The board will be reset if this function is not called before the watchdog timer expires.

**Return** zero(0) or negative error code.

### sensors

Sensor drivers.

#### `bmp280` — BMP280 temperature and pressure sensor


BMP280 is a temperature and pressure sensor from Bosch Sensortec.
This driver supports both I2C and SPI for device communication.

Datasheet: Datasheet BMP280

#### Example usage

This example illustrates how to initialize the driver with an I2C transport layer and then read temperature and pressure from the BMP280 device.

```
struct bmp280_driver_t bmp280;
struct i2c_driver_t i2c;
struct bmp280_transport_i2c_t transport;
float temperature;
float pressure;
float altitude;

/* Initialize and start a I2C driver. */
i2c_init(&i2c, &i2c_device[0], I2C_BAUDRATE_100KBPS, -1);
i2c_start(&i2c);

/* Initialize the BMP280 I2C transport layer. */
bmp280_transport_i2c_init(&transport,
                          &i2c,
                          BMP280_I2C_ADDRESS_AUTOMATIC);
```

(continues on next page)

(continued from previous page)

```
/* Initialize and start the BMP280 driver with the I2C
   transport layer. */
bmp280_init(&bmp280,
            &transport.base,
            bmp280_mode_normal_t,
            bmp280_standby_time_500_us_t,
            bmp280_filter_off_t,
            bmp280_temperature_oversampling_1_t,
            bmp280_pressure_oversampling_1_t);
bmp280_start(&bmp280);

/* Read temperature and pressure from the BMP280. */
bmp280_read(&bmp280, &temperature, &pressure);

/* Calculate the altitude from read pressure. */
altitude = science_pressure_to_altitude(
    pressure,
    SCIENCE_SEA_LEVEL_STANDARD_PRESSURE);

std_printf(OSTR("Temperature: %f\r\n"
                "Pressure: %f\r\n"
                "Altitude: %f\r\n"),
           temperature,
           pressure,
           altitude);
```

---

Source code: src/drivers/sensors/bmp280.h, src/drivers/sensors/bmp280.c

Test code: tst/drivers/software/sensors/bmp280/main.c

Example code: examples/bmp280/main.c

---

### Defines

**BMP280_I2C_ADDRESS_0**
I2C address #0.

**BMP280_I2C_ADDRESS_1**
I2C address #1.

**BMP280_I2C_ADDRESS_AUTOMATIC**
Automatic I2C address detection.

**BMP280_SPI_POLARITY**
Default SPI polarity and phase. Polarity 1 and phase 1 is also supported.

**BMP280_SPI_PHASE**

### Enums

**enum bmp280_mode_t**
Mode configuration.

*Values:*

**bmp280_mode_forced_t** = 1

**bmp280_mode_normal_t** = 3

**enum bmp280_standby_time_t**
Standby time in normal mode configuration.

*Values:*

**bmp280_standby_time_500_us_t** = 0

**bmp280_standby_time_62500_us_t**

**bmp280_standby_time_125_ms_t**

**bmp280_standby_time_250_ms_t**

**bmp280_standby_time_500_ms_t**

**bmp280_standby_time_1_s_t**

**bmp280_standby_time_2_s_t**

**bmp280_standby_time_4_s_t**

**enum bmp280_filter_t**
Filter configuration.

*Values:*

**bmp280_filter_off_t** = 0

**bmp280_filter_2_t**

**bmp280_filter_4_t**

**bmp280_filter_8_t**

**bmp280_filter_16_t**

**enum bmp280_temperature_oversampling_t**
Temperature oversampling configuration.

*Values:*

**bmp280_temperature_off_t** = 0

**bmp280_temperature_oversampling_1_t**

**bmp280_temperature_oversampling_2_t**

**bmp280_temperature_oversampling_4_t**

**bmp280_temperature_oversampling_8_t**

**bmp280_temperature_oversampling_16_t**

**enum bmp280_pressure_oversampling_t**
Pressure oversampling configuration.

*Values:*

**bmp280_pressure_off_t** = 0

**bmp280_pressure_oversampling_1_t**

**bmp280_pressure_oversampling_2_t**

**bmp280_pressure_oversampling_4_t**

**bmp280_pressure_oversampling_8_t**

**bmp280_pressure_oversampling_16_t**

### Functions

int **bmp280_module_init**(void)

Initialize the bmp280 module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **bmp280_init**(**struct** *bmp280_driver_t* *self_p*, **struct** *bmp280_transport_t* *transport_p*, **enum** *bmp280_mode_t* *mode*, **enum** *bmp280_standby_time_t* *standby_time*, **enum** *bmp280_filter_t* *filter*, **enum** *bmp280_temperature_oversampling_t* *temperature_oversampling*, **enum** *bmp280_pressure_oversampling_t* *pressure_oversampling*)

Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `transport_p`: Transport protocol to use. Reference to an I2C or SPI transport object.
- `mode`: Normal or forced mode. In normal mode the device is periodically measuring temperature and pressure. In burst mode the device is in sleeping (low power consumption) and the MCU wakes it to request a measurement.
- `standby_time`: Normal mode standby time.
- `filter`: Filter configuration, normally only set when the device is in normal mode.
- `temperature_oversampling`: Temperature oversampling.
- `pressure_oversampling`: Pressure oversampling.

int **bmp280_start**(**struct** *bmp280_driver_t* *self_p*)

Start given driver by entering normal mode (if mode is `bmp280_mode_normal_t`), and reading calibration data from the device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to start.

int **bmp280_stop**(**struct** *bmp280_driver_t* *self_p*)

Stop given driver by resetting it.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to stop.

int **bmp280_read**(**struct** *bmp280_driver_t* *self_p, float *temperature_p, float *pressure_p)
Read temperature and pressure from the device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `temperature_p`: Temperature in Celsius, or NULL.
- `pressure_p`: Pressure in Pascal, or NULL.

int **bmp280_read_fixed_point**(**struct** *bmp280_driver_t* *self_p, long *temperature_p, long *pressure_p)
Read temperature and pressure from the device and return them as fixed point numbers with three decimal places.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `temperature_p`: Temperature in milli-Celsius, or NULL.
- `pressure_p`: Pressure in milli-Pascal, or NULL.

int **bmp280_transport_i2c_init**(**struct** *bmp280_transport_i2c_t* *self_p, **struct** i2c_driver_t *i2c_p, int *i2c_address*)
Initialize given I2C transport object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: I2C transport object to be initialized.
- `transport_p`: I2C driver to use.
- `i2c_address`: Device I2C address, one of `BMP280_I2C_ADDRESS_0`, `BMP280_I2C_ADDRESS_1` and `BMP280_I2C_ADDRESS_AUTOMATIC`.

int **bmp280_transport_spi_init**(**struct** *bmp280_transport_spi_t* *self_p, **struct** spi_driver_t *spi_p)
Initialize given SPI transport object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SPI transport object to be initialized.
- `spi_p`: SPI driver to use.

**struct bmp280_driver_t**
*#include <bmp280.h>* The BMP280 driver struct.

#### Public Members

**struct** *bmp280_transport_t* ***transport_p**

uint8_t **ctrl_meas**

uint8_t **config**

int16_t **calibration**[12]

**struct** *log_object_t* **log**

**struct bmp280_transport_t**

#### Public Members

**struct** bmp280_transport_protocol_t ***protocol_p**

**struct bmp280_transport_i2c_t**

#### Public Members

**struct** *bmp280_transport_t* **base**

**struct** i2c_driver_t ***i2c_p**

int **i2c_address**

**struct bmp280_transport_spi_t**

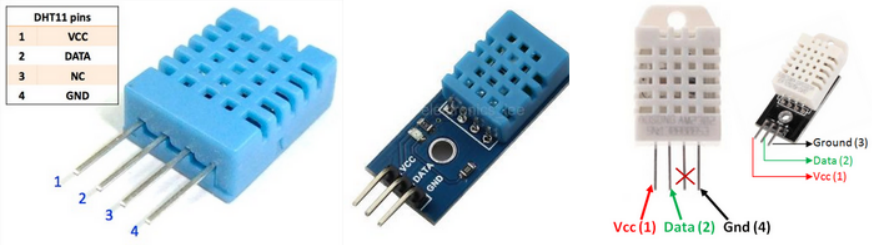
#### Public Members

**struct** *bmp280_transport_t* **base**

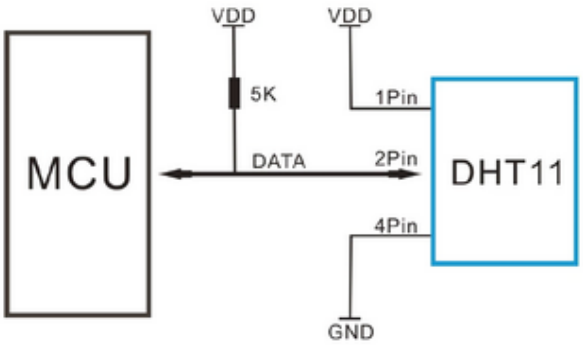
**struct** spi_driver_t ***spi_p**

### dht — DHT temperature and humidity sensor

DHT is temperature and humidity sensor. The two most popular versions are DHT11 (blue) and DHT22 (white).




Both versions are available in bare sensor and module variants. The bare sensor has four pins, while the module only has three. According to the datasheet, the data pin should be connected to VCC via a 5k pull-up resistor. The pull-up resistor is only needed for the bare sen-




sor, as the module already has a built-in pull-up resistor.
Source code: src/drivers/sensors/dht.h, src/drivers/sensors/dht.c

Example code: examples/dht/main.c

---

### Functions

int **dht_module_init**(void)
Initialize the dht module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **dht_init**(**struct** *dht_driver_t* **self_p*, **struct** pin_device_t **pin_p*)
Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `pin_p`: Data line pin device.

int **dht_read**(**struct** *dht_driver_t* **self_p*, float **temperature_p*, float **humidty_p*)
Read temperature and humidity from the DHT21/22 device.

CAUTION: This function disables interrupts for up to 5 ms, which may cause problems for other timing critical functionality.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `temperature_p`: Temperature in degrees Celsius, or NULL.
- `humidity_p`: Humidity in relative humidty RH, or NULL.

int **dht_read_11**(**struct** *dht_driver_t* **self_p*, float **temperature_p*, float **humidty_p*)
Read temperature and humidity from the DHT11 device.

CAUTION: This function disables interrupts for up to 5 ms, which may cause problems for other timing critical functionality.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `temperature_p`: Temperature in degrees Celsius, or NULL.
- `humidity_p`: Humidity in relative humidty RH, or NULL.

**struct dht_driver_t**

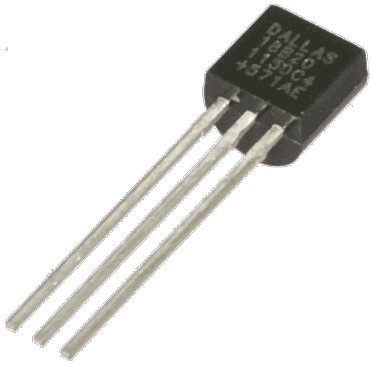
*#include <dht.h>* The DHT driver struct.

#### Public Members

**struct** pin_device_t ***pin_p**

**struct** *log_object_t* **log**

### ds18b20 — One-wire temperature sensor




DS18B20 is a one-wire temperature sensor. Source code: src/drivers/sensors/ds18b20.h, src/drivers/sensors/ds18b20.c

Test code: tst/drivers/hardware/sensors/ds18b20/main.c

---

#### Defines

**DS18B20_FAMILY_CODE**

#### Functions

int **ds18b20_module_init**(void)

Initialize the DS18B20 driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **ds18b20_init**(**struct** *ds18b20_driver_t* *self_p, **struct** *owi_driver_t* *owi_p)
Initialize given driver object. The driver object will communicate with all DS18B20 sensors on given OWI bus.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `owi_p`: One-Wire (OWI) driver.

int **ds18b20_convert**(**struct** *ds18b20_driver_t* *self_p)
Start a temperature convertion on all sensors. The converted temperature can later be read with `ds18b20_read*()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **ds18b20_read**(**struct** *ds18b20_driver_t* *self_p, **const** uint8_t *id_p, float *temperature_p)
Read the most recently converted temperature from given sensor.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `id_p`: Sensor identity.
- `temperature_p`: Measured temperature.

int **ds18b20_read_fixed_point**(**struct** *ds18b20_driver_t* *self_p, **const** uint8_t *id_p, int *temperature_p)
Read the most recently converted temperature from given sensor as a fixed point number.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `id_p`: Sensor identity.
- `temperature_p`: Measured temperature in Q4 fixed point format, or unit 0.0625 degrees Celsius (the raw value read from the sensor).

char *__ds18b20_read_string__(**struct** *ds18b20_driver_t* *self_p, **const** uint8_t *id_p, char *temperature_p)
Read the most recently converted temperature from given sensor as a string.

**Return** `temperature_p` on success, NULL otherwise.

**Parameters**

- `self_p`: Initialized driver object.
- `id_p`: Sensor identity.
- `temperature_p`: Measured temperature as a string.

int **ds18b20_get_temperature**(**struct** *ds18b20_driver_t* *self_p, **const** uint8_t *id_p, int *temperature_p)

Read the most recently converted temperature from given sensor. Call `ds18b20_convert()` to read the temperature from the sensor and update the cached value.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `id_p`: Sensor identity.
- `temperature_p`: Measured temperature in 0.0625 degrees Celsius (the raw value read from the sensor).

char ***ds18b20_get_temperature_str**(**struct** *ds18b20_driver_t* *self_p, **const** uint8_t *id_p, char *temperature_p)

Get temperature for given sensor identity formatted as a string.

**Return** `temperature_p` on success, NULL otherwise.

**Parameters**

- `self_p`: Initialized driver object.
- `id_p`: Sensor identity.
- `temperature_p`: Measured formatted temperature.

**struct ds18b20_driver_t**

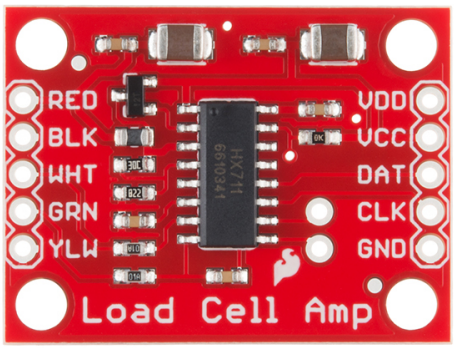
#### Public Members

**struct** *owi_driver_t* ***owi_p**

**struct** *ds18b20_driver_t* ***next_p**

### `hx711` — HX711 ADC for weigh scales

The HX711 chipset performs ADC convertions on weigh scales. It can be used to measure weight in various ranges.




This driver provides methods to read ADC samples from given channel

with given gain.

### Example usage

This is a small example illustrating how to read one sample from each channel and gain combination.

```
struct hx711_driver_t hx711;
float weight_a_128;
float weight_a_64;
float weight_b_32;

/* Initialize and start the deivce. */
hx711_init(&hx711, &pin_d2_dev, &pin_d3_dev, 1.0, 0.0);
hx711_start(&hx711);

/* Read a few samples from the device. */
hx711_read(&hx711, &weight_a_128, hx711_channel_gain_a_128_t);
hx711_read(&hx711, &weight_a_64, hx711_channel_gain_a_64_t);
hx711_read(&hx711, &weight_b_32, hx711_channel_gain_b_32_t);

/* Print the samples. */
std_printf(OSTR("weight_a_128: %f, weight_a_64: %f, weight_b_32: %f\r\n"),
           weight_a_128,
           weight_a_64,
           weight_b_32);

/* Stop the deivce. */
hx711_stop(&hx711);
```

---

Source code: src/drivers/sensors/hx711.h, src/drivers/sensors/hx711.c

Test code: tst/drivers/software/sensors/hx711/main.c

Test coverage: src/drivers/sensors/hx711.c

Example code: examples/hx711/main.c

---

### Enums

**enum hx711_channel_gain_t**

*Values:*

**hx711_channel_gain_a_128_t** = 1

**hx711_channel_gain_b_32_t**

**hx711_channel_gain_a_64_t**

### Functions

int **hx711_module_init**(void)

Initialize the hx711 module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **hx711_init**(**struct** *hx711_driver_t* *self_p, **struct** pin_device_t *pd_sck_p, **struct** pin_device_t *dout_p, float *scale*, float *offset*)
Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `pd_sck_p`: PD_SCK pin device.
- `dout_p`: DOUT pin device.
- `scale`: Scale value to multiply with read samples after the offset has been added.
- `offset`: Offset value to add to read samples before they are scaled.

int **hx711_start**(**struct** *hx711_driver_t* *self_p)
Start the driver by configuring the pins and resetting the HX711.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to start.

int **hx711_stop**(**struct** *hx711_driver_t* *self_p)
Stop given driver by setting pins as input.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to stop.

int **hx711_read**(**struct** *hx711_driver_t* *self_p, float *weight_p, **enum** *hx711_channel_gain_t* *channel_gain*)
Read a offsetted and scaled weight from given channel and gain combination.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `weight_p`: Measured offsetted and scaled weight.
- `channel_gain`: Channel and gain combination.

int **hx711_read_raw**(**struct** *hx711_driver_t* *self_p, int32_t *sample_p, **enum** *hx711_channel_gain_t* *channel_gain*)
Read a sample from given channel and gain combination and output the sign extended raw read value. No offsetting or scaling is performed.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

- `sample_p`: Sign extended read sample.
- `channel_gain`: Channel and gain combination.

int **hx711_set_scale**(**struct** *hx711_driver_t* *self_p, float *scale*)
Set the scale value.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `scale`: Scale value to multiply with read samples after the offset has been added.

int **hx711_set_offset**(**struct** *hx711_driver_t* *self_p, float *offset*)
Set the offset value.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `offset`: Offset value to add to read samples before they are scaled.

**struct hx711_driver_t**

#### Public Members

**struct** pin_device_t ***pd_sck_p**

**struct** pin_device_t ***dout_p**

float **scale**

float **offset**

### sht3xd — SHT3x-D Humidity and Temperature Sensor


The Sensirion SHT3x-D is a series of digital of Humidity and Temperature Sensors. This driver supports the SHT30-D, SHT31-D, and SHT35-D using an I2C interface. The analog SHT3x-A, such as SHT30-A and SHT31-A are not supported.

The SHT3x-D sensors supports I2C speed of up to 1MHz.

Current limitations of this driver:

- Only supports basic functionality and high repeatability mode.

- Does not perform check CRC of sensor result.

Datasheet: Datasheet SHT3x-DIS

Source code: src/drivers/sensors/sht3xd.h, src/drivers/sensors/sht3xd.c

---

### Defines

**SHT3X_DIS_I2C_ADDR_A**
SHT3x-DIS default I2C address.

**SHT3X_DIS_I2C_ADDR_B**
SHT3x-DIS alternate I2C address.

**MEASUREMENT_DURATION_HIGH_MS**
Max measurement time for high repeatability.

### Functions

int **sht3xd_module_init**(void)
Initialize the driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **sht3xd_init**(**struct** *sht3xd_driver_t* **self_p*, **struct** i2c_driver_t **i2c_p*, int *i2c_addr*)
Initialize driver object. The driver object will be used for a single sensor.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialize.
- `i2c_p`: The I2C driver pointer.
- `i2c_addr`: The address of the SHT3x-D. Probably SHT3X_DIS_I2C_ADDR_A.

int **sht3xd_start**(**struct** *sht3xd_driver_t* **self_p*)
Start the driver.

This verify the sensor is present.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **sht3xd_get_temp_humid**(**struct** *sht3xd_driver_t* **self_p*, float **temp_p*, float **humid_p*)
Get measurements and return it from the SHD3x-DIS chip.

This is a "high level" function which will block for the time it takes the sensor to perform the measurement.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `temp_p`: Tempererature in Celsius, or NULL.
- `humid_p`: Relative Humidity, or NULL.

int **sht3xd_get_serial**(**struct** *sht3xd_driver_t* *self_p, uint32_t *serial_p)

Get the serial number from the SHD3x-D.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `serial_p`: Serial number of the SHT3x-D sensor.

**struct sht3xd_driver_t**

#### Public Members

**struct** i2c_driver_t ***i2c_p**

int **i2c_addr**

uint32_t **serial**

### storage

Persistent storage drivers.

### eeprom_i2c — I2C EEPROM

Below is a list of I2C EEPROMs that are known to work with this driver. Other I2C EEPROMs may work as well, as they often implement the same interface.

- AT24C32 from Atmel.
- AT24C256 from Atmel.

Known limitations:

- Only supports 16 bits addressing. 8 bits addressing can easily be added.

---

Source code: src/drivers/storage/eeprom_i2c.h, src/drivers/storage/eeprom_i2c.c

Test code: tst/drivers/hardware/storage/eeprom_i2c/main.c

---

### Functions

int **eeprom_i2c_module_init**(void)

Initialize EEPROM I2C module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **eeprom_i2c_init**(**struct** *eeprom_i2c_driver_t* **self_p*, **struct** i2c_driver_t **i2c_p*, int *i2c_address*, uint32_t *size*)

Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `i2c_p`: Initialized and started I2C driver object.
- `i2c_address`: The I2C address of the EEPROM.
- `size`: Size of the EEPROM in bytes.

ssize_t **eeprom_i2c_read**(**struct** *eeprom_i2c_driver_t* **self_p*, void **dst_p*, uint32_t *src*, size_t *size*)

Read into given buffer from given EEPROM address.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst_p`: Buffer to read into.
- `src`: EEPROM address to read from.
- `size`: Number of bytes to read.

ssize_t **eeprom_i2c_write**(**struct** *eeprom_i2c_driver_t* **self_p*, uint32_t *dst*, **const** void **src_p*, size_t *size*)

Write given buffer to given EEPROM address.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst`: EEPROM address to write to.
- `src_p`: Buffer to write.
- `size`: Number of bytes to write.

**struct eeprom_i2c_driver_t**

### Public Members

**struct** i2c_driver_t ***i2c_p**

int **i2c_address**

uint32_t **size**

## eeprom_soft — Software EEPROM

Source code: src/drivers/storage/eeprom_soft.h, src/drivers/storage/eeprom_soft.c

Test code: tst/drivers/hardware/storage/eeprom_soft/main.c

---

### Functions

int **eeprom_soft_module_init**(void)

Initialize software EEPROM module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **eeprom_soft_init**(**struct** *eeprom_soft_driver_t* **self_p*, **struct** flash_driver_t **flash_p*, **const struct** *eeprom_soft_block_t* **blocks_p*, int *number_of_blocks*, size_t *chunk_size*)

Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `flash_p`: Flash driver.
- `blocks_p`: Flash memory blocks to use.
- `number_of_blocks`: Number of blocks.
- `chunk_size`: Chunk size in bytes. This is the size of the EEPROM. Eight bytes of the chunk will be used to store metadata, so only `chunk_size - 8` bytes are available to the user.

int **eeprom_soft_mount**(**struct** *eeprom_soft_driver_t* **self_p*)

Mount given software EEPROM.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to mount.

int **eeprom_soft_format**(**struct** *eeprom_soft_driver_t* **self_p*)

Format given software EEPROM.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to format.

ssize_t **eeprom_soft_read**(**struct** *eeprom_soft_driver_t* *self_p, void *dst_p, uintptr_t src, size_t size)
Read into given buffer from given address.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst_p`: Buffer to read into.
- `src`: Software EEPROM address to read from. Addressing starts at zero(0).
- `size`: Number of bytes to read.

ssize_t **eeprom_soft_write**(**struct** *eeprom_soft_driver_t* *self_p, uintptr_t dst, **const** void *src_p, size_t size)
Write given buffer to given address.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst`: Software EEPROM address to write to. Addressing starts at zero(0).
- `src_p`: Buffer to write.
- `size`: Number of bytes to write.

ssize_t **eeprom_soft_vwrite**(**struct** *eeprom_soft_driver_t* *self_p, **struct** *iov_uintptr_t* *dst_p, **struct** *iov_t* *src_p, size_t *length*)
Write given buffers to given NVM addresses.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst_p`: Software EEPROM address ranges to write, sorted from lowest to highest. Addressing starts at zero(0).
- `src_p`: Buffers to write in the same order as in `dst_p`. The size fields are not used.
- `length`: Number of elements in `dst_p` and `src_p`.

**struct eeprom_soft_block_t**

### Public Members

uintptr_t **address**

size_t **size**

**struct eeprom_soft_driver_t**

### Public Members

**struct** flash_driver_t ***flash_p**

**const struct** *eeprom_soft_block_t* ***blocks_p**

int **number_of_blocks**

size_t **chunk_size**

size_t **eeprom_size**

**const struct** *eeprom_soft_block_t* ***block_p**

uintptr_t **chunk_address**

uint16_t **revision**

**struct eeprom_soft_driver_t::@41 eeprom_soft_driver_t::current**

## `flash` — Flash memory

Source code: src/drivers/storage/flash.h, src/drivers/storage/flash.c

Test code: tst/drivers/hardware/storage/flash/main.c

---

### Functions

int **flash_module_init**(void)
Initialize the flash module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **flash_init**(**struct** flash_driver_t **self_p*, **struct** flash_device_t **dev_p*)
Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `dev_p`: Device to use.

ssize_t **flash_read**(**struct** flash_driver_t **self_p*, void **dst_p*, uintptr_t *src*, size_t *size*)
Read data from given address in given flash memory.

**Return** Number of read bytes `size`, or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst_p`: Buffer to read into.
- `src`: Address in flash memory to read from.

- `size`: Number of bytes to receive.

ssize_t **flash_write**(**struct** flash_driver_t *self_p, uintptr_t *dst*, **const** void *src_p, size_t *size*)
Write data to given address in given flash memory.

**Return** Number of written bytes `size`, or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst`: Address in flash memory to write to.
- `src_p`: Buffer to write.
- `size`: Number of bytes to write.

int **flash_erase**(**struct** flash_driver_t *self_p, uintptr_t *addr*, size_t *size*)
Erase all sectors part of given memory range.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst`: Address in flash memory to erase from.
- `size`: Number of bytes to erase.

### Variables

**struct** flash_device_t **flash_device**[**FLASH_DEVICE_MAX**]

### `sd` — Secure Digital memory

Source code: src/drivers/storage/sd.h, src/drivers/storage/sd.c

Test code: tst/drivers/hardware/storage/sd/main.c

---

### Defines

**SD_ERR_NORESPONSE_WAIT_FOR_DATA_START_BLOCK**

**SD_ERR_GO_IDLE_STATE**

**SD_ERR_CRC_ON_OFF**

**SD_ERR_SEND_IF_COND**

**SD_ERR_CHECK_PATTERN**

**SD_ERR_SD_SEND_OP_COND**

**SD_ERR_READ_OCR**

**SD_ERR_READ_COMMAND**

**SD_ERR_READ_DATA_START_BLOCK**

**SD_ERR_READ_WRONG_DATA_CRC**

**SD_ERR_WRITE_BLOCK**

**SD_ERR_WRITE_BLOCK_TOKEN_DATA_RES_ACCEPTED**

**SD_ERR_WRITE_BLOCK_WAIT_NOT_BUSY**

**SD_ERR_WRITE_BLOCK_SEND_STATUS**

**SD_BLOCK_SIZE**

**SD_CCC**(csd_p)

**SD_C_SIZE**(csd_p)

**SD_C_SIZE_MULT**(csd_p)

**SD_SECTOR_SIZE**(csd_p)

**SD_WRITE_BL_LEN**(csd_p)

**SD_CSD_STRUCTURE_V1**

**SD_CSD_STRUCTURE_V2**

### Functions

int **sd_init**(**struct** *sd_driver_t* *self_p, **struct** spi_driver_t *spi_p)
Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.

int **sd_start**(**struct** *sd_driver_t* *self_p)
Start given SD card driver. This resets the SD card and performs the initialization sequence.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **sd_stop**(**struct** *sd_driver_t* *self_p)
Stop given SD card driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

ssize_t **sd_read_cid**(**struct** *sd_driver_t* *self_p, **struct** *sd_cid_t* *cid_p)
Read card CID register. The CID contains card identification information such as Manufacturer ID, Product name, Product serial number and Manufacturing date.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `cid`: pointer to cid data store.

ssize_t **sd_read_csd**(**struct** *sd_driver_t* *self_p, **union** *sd_csd_t* *csd_p)
Read card CSD register. The CSD contains that provides information regarding access to the card's contents.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `csd`: pointer to csd data store.

ssize_t **sd_read_block**(**struct** *sd_driver_t* *self_p, void *dst_p, uint32_t src_block)
Read given block from SD card.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `src_block`: Block to read from.

ssize_t **sd_write_block**(**struct** *sd_driver_t* *self_p, uint32_t dst_block, **const** void *src_p)
Write data to the SD card.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `dst_block`: Block to write to.
- `src_p`: Buffer to write.

## Variables

**struct** *sd_csd_v2_t* **PACKED**

**struct sd_cid_t**

### Public Members

uint8_t **mid**

char **oid**[2]

char **pnm**[5]

uint8_t **prv**

uint32_t **psn**

uint16_t **mdt**

uint8_t **crc**

**struct sd_csd_v1_t**

### Public Members

uint8_t **reserved1**

uint8_t **csd_structure**

uint8_t **taac**

uint8_t **nsac**

uint8_t **tran_speed**

uint8_t **ccc_high**

uint8_t **read_bl_len**

uint8_t **ccc_low**

uint8_t **c_size_high**

uint8_t **reserved2**

uint8_t **dsr_imp**

uint8_t **read_blk_misalign**

uint8_t **write_blk_misalign**

uint8_t **read_bl_partial**

uint8_t **c_size_mid**

uint8_t **vdd_r_curr_max**

uint8_t **vdd_r_curr_min**

uint8_t **c_size_low**

uint8_t **c_size_mult_high**

uint8_t **vdd_w_curr_max**

uint8_t **vdd_w_curr_min**

uint8_t **sector_size_high**

uint8_t **erase_blk_en**

uint8_t **c_size_mult_low**

uint8_t **wp_grp_size**

uint8_t **sector_size_low**

uint8_t **write_bl_len_high**

uint8_t **r2w_factor**

uint8_t **reserved3**

uint8_t **wp_grp_enable**

uint8_t **reserved4**

uint8_t **write_bl_partial**

uint8_t **write_bl_len_low**

uint8_t **reserved5**

uint8_t **file_format**

uint8_t **tmp_write_protect**

uint8_t **perm_write_protect**

uint8_t **copy**

uint8_t **file_format_grp**

uint8_t **crc**

**struct sd_csd_v2_t**

### Public Members

uint8_t **reserved1**

uint8_t **csd_structure**

uint8_t **taac**

uint8_t **nsac**

uint8_t **tran_speed**

uint8_t **ccc_high**

uint8_t **read_bl_len**

uint8_t **ccc_low**

uint8_t **reserved2**

uint8_t **dsr_imp**

uint8_t **read_blk_misalign**

uint8_t **write_blk_misalign**

uint8_t **read_bl_partial**

uint8_t **c_size_high**

uint8_t **reserved3**

uint8_t **c_size_mid**

uint8_t **c_size_low**

uint8_t **sector_size_high**

uint8_t **erase_blk_en**

uint8_t **reserved4**

uint8_t **wp_grp_size**

uint8_t **sector_size_low**

uint8_t **write_bl_len_high**

uint8_t **r2w_factor**

uint8_t **reserved5**

uint8_t **wp_grp_enable**

uint8_t **reserved6**

uint8_t **write_bl_partial**

uint8_t **write_bl_len_low**

uint8_t **reserved7**

uint8_t **file_format**

uint8_t **tmp_write_protect**

uint8_t **perm_write_protect**

uint8_t **copy**

uint8_t **file_format_grp**

uint8_t **crc**

**union sd_csd_t**

#### Public Members

**struct** *sd_csd_v1_t* **v1**

**struct** *sd_csd_v2_t* **v2**

**struct sd_driver_t**

#### Public Members

**struct** spi_driver_t ***spi_p**

int **type**

### network

Networking drivers.

### can — Controller Area Network

A Controller Area Network (CAN bus) is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based protocol, designed originally for multiplex electrical wiring within automobiles, but is also used in many other contexts.

Below is a short example of how to use this module. The error handling is left out for readability.

```
struct can_frame_t can_rx_buf[8];
struct can_frame_t frame;

/* Initialize and start the CAN conroller. */
can_init(&can,
         &can_device[0],
```

(continues on next page)

(continued from previous page)

```
        CAN_SPEED_500KBPS,
        can_rx_buf,
        sizeof(can_rx_buf)) == 0);
can_start(&can);

/* Read a frame from the bus. */
can_read(&can, &frame, sizeof(frame));

/* Stop the CAN controller. */
can_stop(&can);
```

---

Source code: src/drivers/network/can.h, src/drivers/network/can.c

Test code: tst/drivers/hardware/network/network/can/main.c

---

### Defines

**CAN_SPEED_1000KBPS**

**CAN_SPEED_500KBPS**

**CAN_SPEED_250KBPS**

### Functions

int **can_module_init**(void)

Initialize CAN module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **can_init**(**struct** can_driver_t *self_p, **struct** can_device_t *dev_p, uint32_t *speed*, void *rxbuf_p, size_t *size*)

Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `dev_p`: CAN device to use.
- `speed`: Can bus speed. One of the defines with the prefix`CAN_SPEED_`.
- `rxbuf_p`: CAN frame reception buffer.
- `size`: Size of the reception buffer in bytes.

int **can_start**(**struct** can_driver_t *self_p)

Starts the CAN device using configuration in given driver object.

**Return** zero(0) or negative error code. Value -ENETDOWN indicates a problem with connection to the bus and probably to the transceiver.

**Parameters**

- `self_p`: Initialized driver object.

int **can_stop**(**struct** can_driver_t *_self_p_)
Stops the CAN device referenced by given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

ssize_t **can_read**(**struct** can_driver_t *_self_p_, **struct** _can_frame_t *frame_p_, size_t _size_)
Read one or more CAN frames from the CAN bus. Blocks until the frame(s) are received.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `frame_p`: Array of read frames.
- `size`: Size of frames buffer in bytes. Must be a multiple of `sizeof(struct can_frame_t)`.

ssize_t **can_write**(**struct** can_driver_t *_self_p_, **const struct** _can_frame_t *frame_p_, size_t _size_)
Write one or more CAN frames to the CAN bus. Blocks until the frame(s) have been transmitted.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `frame_p`: Array of frames to write.
- `size`: Size of frames buffer in bytes. Must be a multiple of `sizeof(struct can_frame_t)`.

### Variables

**struct** can_device_t **can_device**[**CAN_DEVICE_MAX**]

**struct can_frame_t**

#### Public Members

uint32_t **id**

uint8_t **extended_frame**

uint8_t **rtr**

uint8_t **size**

**struct can_frame_t::@1 can_frame_t::@2**

uint8_t **u8**[8]

uint32_t **u32**[2]

**union can_frame_t::@3 can_frame_t::data**

### esp_wifi — Espressif WiFi

This module is a wrapper for the Espressif WiFi interface.

Configure the WiFi as a Station and an Access Point at the same time. The application tries to connect to a Wifi with SSID *ssid* and will accept connections to the SSID *Simba*.

```
esp_wifi_set_op_mode(esp_wifi_op_mode_station_softap_t);
esp_wifi_softap_init("Simba", NULL);
esp_wifi_station_init("ssid", "password", NULL, NULL);
```

Configure the WiFi as an Access Point. The application will accept connections to the SSID *Simba*.

```
esp_wifi_set_op_mode(esp_wifi_op_mode_softap_t);
esp_wifi_softap_init("Simba", NULL);
```

Configure the WiFi as a Station. The application tries to connect to a Wifi with SSID *ssid*.

```
esp_wifi_set_op_mode(esp_wifi_op_mode_station_t);
esp_wifi_station_init("ssid", "password", NULL, NULL);
```

Configure the WiFi as a Station specifying the MAC address of the access point. The application tries to connect to a Wifi with a MAC of *c8:d7:19:0f:04:66* and SSID *ssid*.

```
esp_wifi_set_op_mode(esp_wifi_op_mode_station_t);
esp_wifi_station_init("ssid",
                      "password",
                      (uint8_t[]){0xc8, 0xd7, 0x19, 0x0f, 0x04, 0x66},
                      NULL);
```

---

Submodules:

### esp_wifi_softap — Espressif WiFi SoftAP

This module is a wrapper for the Espressif WiFi SoftAP interface.

---

Source code: src/drivers/network/esp_wifi/softap.h, src/drivers/network/esp_wifi/softap.c

Test code: tst/drivers/hardware/network/esp_wifi/softap/main.c

---

## Functions

int **esp_wifi_softap_init**(**const** char *_ssid_p_, **const** char *_password_p_)

Initialize the WiFi SoftAP interface.

**Return** zero(0) or negative error code.

**Parameters**

- `ssid_p`: SSID of the SoftAP.
- `password_p`: Password of SoftAP.

int **esp_wifi_softap_set_ip_info**(**const struct** *inet_if_ip_info_t* *info_p)
Set the ip address, netmask and gateway of the WiFi SoftAP.

**Return** zero(0) or negative error code.

int **esp_wifi_softap_get_ip_info**(**struct** *inet_if_ip_info_t* *info_p)
Get the SoftAP ip address, netmask and gateway.

**Return** zero(0) or negative error code.

**Parameters**

- `info_p`: Read ip information.

int **esp_wifi_softap_get_number_of_connected_stations**(void)
Get the number of stations connected to the SoftAP.

**Return** Number of conencted stations.

int **esp_wifi_softap_get_station_info**(**struct** *esp_wifi_softap_station_info_t* *info_p, int *length*)
Get the information of stations connected to the SoftAP, including MAC and IP addresses.

**Return** Number of valid station information entries or negative error code.

**Parameters**

- `info_p`: An array to write the station information to.
- `length`: Length of the info array.

int **esp_wifi_softap_dhcp_server_start**(void)
Enable the SoftAP DHCP server.

**Return** zero(0) or negative error code.

int **esp_wifi_softap_dhcp_server_stop**(void)
Disable the SoftAP DHCP server. The DHCP server is enabled by default.

**Return** zero(0) or negative error code.

**enum** *esp_wifi_dhcp_status_t* **esp_wifi_softap_dhcp_server_status**(void)
Get the SoftAP DHCP server status.

**Return** DHCP server status.

**struct esp_wifi_softap_station_info_t**
*#include <softap.h>* Information about a connected station.

### Public Members

uint8_t **bssid**[6]

**struct** *inet_ip_addr_t* **ip_address**

## esp_wifi_station — Espressif WiFi Station

This module is a wrapper for the Espressif WiFi station interface.

---

Source code: src/drivers/network/esp_wifi/station.h, src/drivers/network/esp_wifi/station.c

Test code: tst/drivers/hardware/network/esp_wifi/station/main.c

---

### Enums

**enum esp_wifi_station_status_t**

WiFi station connection status.

*Values:*

**esp_wifi_station_status_idle_t** = 0

**esp_wifi_station_status_connecting_t**

**esp_wifi_station_status_auth_failure_t**

**esp_wifi_station_status_no_ap_found_t**

**esp_wifi_station_status_connect_fail_t**

**esp_wifi_station_status_got_ip_t**

**esp_wifi_station_status_connected_t**

### Functions

int **esp_wifi_station_init**(**const** char **ssid_p*, **const** char **password_p*, **const** uint8_t **bssid_p*, **const struct** *inet_if_ip_info_t* **info_p*)

Initialize the WiFi station.

**Return** zero(0) or negative error code.

**Parameters**

- `ssid_p`: WiFi SSID to connect to.
- `password_p`: WiFi password.
- `bssid_p`: WiFi station MAC (BSSID) or NULL to ignore.
- `info_p`: Static ip configration or NULL to use DHCP.

int **esp_wifi_station_connect**(void)

Connect the WiFi station to the Access Point (AP).

**Return** zero(0) or negative error code.

int **esp_wifi_station_disconnect**(void)
Disconnect the WiFi station from the AP.

**Return** zero(0) or negative error code.

int **esp_wifi_station_set_ip_info**(**const struct** *inet_if_ip_info_t* **info_p*)
Set the ip address, netmask and gateway of the WiFi station.

**Return** zero(0) or negative error code.

int **esp_wifi_station_get_ip_info**(**struct** *inet_if_ip_info_t* **info_p*)
Get the station ip address, netmask and gateway.

**Return** zero(0) or negative error code.

int **esp_wifi_station_set_reconnect_policy**(int *policy*)
Set whether the station will reconnect to the AP after disconnection. It will do so by default.

**Return** zero(0) or negative error code.

**Parameters**

- `policy`: If it's true, it will enable reconnection; if it's false, it will disable reconnection.

int **esp_wifi_station_get_reconnect_policy**(void)
Check whether the station will reconnect to the AP after disconnection.

**Return** true(1) or false(0).

**enum** *esp_wifi_station_status_t* **esp_wifi_station_get_status**(void)
Get the connection status of the WiFi station.

**Return** The connection status.

int **esp_wifi_station_dhcp_client_start**(void)
Enable the station DHCP client.

**Return** zero(0) or negative error code.

int **esp_wifi_station_dhcp_client_stop**(void)
Disable the station DHCP client.

**Return** zero(0) or negative error code.

**enum** *esp_wifi_dhcp_status_t* **esp_wifi_station_dhcp_client_status**(void)
Get the station DHCP client status.

**Return** Station DHCP client status.

**const** char ***esp_wifi_station_status_as_string**(**enum** *esp_wifi_station_status_t status*)
Convert given status code to a string.

**Return** Status code as a string.

Source code: src/drivers/network/esp_wifi.h, src/drivers/network/esp_wifi.c

Test code: tst/drivers/hardware/network/esp_wifi/main.c

## Enums

**enum esp_wifi_op_mode_t**
WiFi operational mode.

**Note** The driver code rely on the combined station + softap mode being bitwise-or of station and softap mode, and the enum integer values of the modes matching the ESP8266 SDKs values for the modes.

*Values:*

**esp_wifi_op_mode_null_t** = 0

**esp_wifi_op_mode_station_t**

**esp_wifi_op_mode_softap_t**

**esp_wifi_op_mode_station_softap_t**

**esp_wifi_op_mode_max_t**

**enum esp_wifi_phy_mode_t**
Physical WiFi mode.

*Values:*

**esp_wifi_phy_mode_11b_t** = 1

**esp_wifi_phy_mode_11g_t**

**esp_wifi_phy_mode_11n_t**

**enum esp_wifi_dhcp_status_t**
DHCP status.

*Values:*

**esp_wifi_dhcp_status_stopped_t** = 0

**esp_wifi_dhcp_status_running_t**

## Functions

int **esp_wifi_module_init**(void)
Initialize the Espressif WiFi module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **esp_wifi_set_op_mode**(**enum** *esp_wifi_op_mode_t mode*)
Set the WiFi operating mode to None, Station, SoftAP or Station + SoftAP. The default mode is SoftAP.

**Return** zero(0) or negative error code.

**Parameters**

- `mode`: Operating mode to set.

**enum** *esp_wifi_op_mode_t* **esp_wifi_get_op_mode**(void)
Get the current WiFi operating mode. The operating mode can be None, Station, SoftAP, or Station + SoftAP.

**Return** Current operating mode.

int **esp_wifi_set_phy_mode**(**enum** *esp_wifi_phy_mode_t mode*)
Set the WiFi physical mode (802.11b/g/n).

The SoftAP only supports b/g.

**Return** zero(0) or negative error code.

**Parameters**

- `mode`: Physical mode.

**enum** *esp_wifi_phy_mode_t* **esp_wifi_get_phy_mode**(void)
Get the physical mode (802.11b/g/n).

**Return** WiFi physical mode.

void **esp_wifi_print**(void **chout_p*)
Print information about the WiFi.

### i2c — I2C

I2C is a data transfer bus. Normally one master and one or more slaves are connected to the bus. The master addresses one slave at a time to transfer data between the devices.

The master is normally fairly easy to implement since it controls the bus clock and no race conditions can occur. The slave, on the other hand, can be implemented in various ways depending on the application requirements. In this implementation the slave will always send an acknowledgement when addressed by the master, and lock the bus by pulling SCL low until it is ready for the transmission.

For systems without hardware I2C, the *i2c_soft — Software I2C* driver will supply I2C devices though the interface documented in this driver.

---

Source code: src/drivers/network/i2c.h, src/drivers/network/i2c.c

Test code: tst/drivers/hardware/network/i2c/master/main.c

---

#### Defines

**I2C_BAUDRATE_3_2MBPS**

**I2C_BAUDRATE_1MBPS**

**I2C_BAUDRATE_400KBPS**

**I2C_BAUDRATE_100KBPS**

### Functions

int **i2c_module_init**(void)

Initialize the i2c module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **i2c_init**(**struct** i2c_driver_t *self_p, **struct** i2c_device_t *dev_p, int *baudrate*, int *address*)

Initialize given driver object. The same driver object is used for both master and slave modes. Use `i2c_start()` to start the device as a master, and `i2c_slave_start()` to start it as a slave.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `dev_p`: I2C device to use.
- `baudrates`: Bus baudrate when in master mode. Unused in slave mode.
- `address`: Slave address when in slave mode. Unused in master mode.

int **i2c_start**(**struct** i2c_driver_t *self_p)

Start given driver object in master mode. Enables data reception and transmission, but does not start any transmission. Use `i2c_read()` and `i2c_write()` to exchange data with the peer.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.

int **i2c_stop**(**struct** i2c_driver_t *self_p)

Stop given driver object. Disables data reception and transmission in master mode.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.

ssize_t **i2c_read**(**struct** i2c_driver_t *self_p, int *address*, void *buf_p, size_t *size*)

Read given number of bytes into given buffer from given slave.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Driver object.
- `address`: Slave address to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **i2c_write**(**struct** i2c_driver_t *self_p, int *address*, **const** void *buf_p, size_t *size*)

Write given number of bytes from given buffer to given slave.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Driver object.
- `address`: Slave address to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

int **i2c_scan**(**struct** i2c_driver_t *_self_p_, int _address_)
Scan the i2c bus for a slave with given address.

**Return** true(1) if a slave responded to given address, otherwise false(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `address`: Address of the slave to scan for.

int **i2c_slave_start**(**struct** i2c_driver_t *_self_p_)
Start given driver object in slave mode. Enables data reception and transmission, but does not start any transmission. Data transfers are started by calling the `i2c_slave_read()` and `i2c_slave_write()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.

int **i2c_slave_stop**(**struct** i2c_driver_t *_self_p_)
Stop given driver object. Disables data reception and transmission in slave mode.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.

ssize_t **i2c_slave_read**(**struct** i2c_driver_t *_self_p_, void *_buf_p_, size_t _size_)
Read into given buffer from the next master that addresses this slave.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **i2c_slave_write**(**struct** i2c_driver_t *_self_p_, **const** void *_buf_p_, size_t _size_)
Write given buffer to the next master that addresses this slave.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Driver object.

- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

### Variables

**struct** i2c_device_t **i2c_device**[**I2C_DEVICE_MAX**]

## `i2c_soft` — Software I2C

I2C is a data transfer bus. Normally one master and one or more slaves are connected to the bus. The master addresses one slave at a time to transfer data between the devices.

This driver implements I2C in software for MCUs without I2C hardware support. For systems with hardware I2C support, the *i2c — I2C* driver will probably be preferable.

---

Source code: src/drivers/network/i2c_soft.h, src/drivers/network/i2c_soft.c

Test code: tst/drivers/hardware/network/i2c/master_soft/main.c

---

### Functions

int **i2c_soft_module_init**(void)

Initialize the i2c soft module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **i2c_soft_init**(**struct** *i2c_soft_driver_t* **self_p*, **struct** pin_device_t **scl_dev_p*, **struct** pin_device_t **sda_dev_p*, long *baudrate*, long *max_clock_stretching_us*, long *clock_stretching_sleep_us*)

Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `scl_dev_p`: The I2C clock pin (SCL).
- `sda_dev_p`: The I2C data pin (SDA).
- `baudrate`: Bus baudrate.
- `max_clock_stretching_us`: Maximum number of microseconds to wait for the clock stretching to end.
- `clock_stretching_sleep_us`: SCL poll interval in number of microseconds waiting for clock stretching to end.

int **i2c_soft_start**(**struct** *i2c_soft_driver_t* *self_p)
Start given driver object. Enables data reception and transmission, but does not start any transmission. Data transfers are started by calling the `i2c_soft_read()` and `i2c_soft_write()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.

int **i2c_soft_stop**(**struct** *i2c_soft_driver_t* *self_p)
Stop given driver object. Disables data reception and transmission.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.

ssize_t **i2c_soft_read**(**struct** *i2c_soft_driver_t* *self_p, int *address*, void *buf_p, size_t *size*)
Read given number of bytes into given buffer from given slave.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Driver object.
- `address`: Slave address to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **i2c_soft_write**(**struct** *i2c_soft_driver_t* *self_p, int *address*, **const** void *buf_p, size_t *size*)
Write given number of bytes from given buffer to given slave.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Driver object.
- `address`: Slave address to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

int **i2c_soft_scan**(**struct** *i2c_soft_driver_t* *self_p, int *address*)
Scan the i2c bus for a slave with given address.

**Return** true(1) if a slave responded to given address, otherwise false(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `address`: Address of the slave to scan for.

**struct i2c_soft_driver_t**

### Public Members

**struct** pin_device_t ***scl_p**

**struct** pin_device_t ***sda_p**

long **baudrate**

long **baudrate_us**

long **max_clock_stretching_us**

long **clock_stretching_sleep_us**

## icsp_soft — Software ICSP

---

Source code: src/drivers/network/icsp_soft.h, src/drivers/network/icsp_soft.c

Test code: tst/drivers/software/network/icsp_soft/master/main.c

---

### Functions

int **icsp_soft_module_init**(void)

Initialize ICSP soft module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **icsp_soft_init**(**struct** *icsp_soft_driver_t* **self_p*, **struct** pin_device_t **pgec_p*, **struct** pin_device_t **pged_p*, **struct** pin_device_t **mclrn_p*)

Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `pgec_p`: Clock.
- `pged_p`: Data.
- `mclrn_p`: Master control (reset).

int **icsp_soft_start**(**struct** *icsp_soft_driver_t* **self_p*)

Start given ICSP soft driver. Configures all pins and enters the idle state in the ICSP TAP controller state machine.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **icsp_soft_stop**(**struct** *icsp_soft_driver_t *self_p*)

Stop given ICSP soft driver. Enters the reset state in the ICSP TAP controller state machine and then configures all pins as inputs.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **icsp_soft_reset**(**struct** *icsp_soft_driver_t *self_p*)

Reset the ICSP device and then enter the idle state.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **icsp_soft_instruction_write**(**struct** *icsp_soft_driver_t *self_p*, **const** void *buf_p, size_t *number_of_bits*)

Write given instruction to given ICSP device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Instruction to write.
- `size`: Number of bits to write.

int **icsp_soft_data_transfer**(**struct** *icsp_soft_driver_t *self_p*, void *rxbuf_p, **const** void *txbuf_p, size_t *number_of_bits*)

Simultaniuos data read/write operation from/to given ICSP device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `rxbuf_p`: Buffer to read into.
- `txbuf_p`: Buffer to write.
- `size`: Number of bits to transfer.

int **icsp_soft_data_read**(**struct** *icsp_soft_driver_t *self_p*, void *buf_p, size_t *number_of_bits*)

Read data from given ICSP device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bits to read.

int **icsp_soft_data_write**(**struct** *icsp_soft_driver_t* *self_p, **const** void *buf_p, size_t *number_of_bits*)

Write given data to the ICSP device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to write.
- `size`: Number of bits to write.

int **icsp_soft_fast_data_transfer**(**struct** *icsp_soft_driver_t* *self_p, void *rxbuf_p, **const** void *txbuf_p, size_t *number_of_bits*)

Simultaniuos fast data read/write operation from/to given ICSP device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `rxbuf_p`: Buffer to read into.
- `txbuf_p`: Buffer to write.
- `size`: Number of bits to transfer.

int **icsp_soft_fast_data_read**(**struct** *icsp_soft_driver_t* *self_p, uint32_t *data_p*)

Fast read data from given ICSP device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `data_p`: Read data.

int **icsp_soft_fast_data_write**(**struct** *icsp_soft_driver_t* *self_p, uint32_t *data*)

Fast write given data to the ICSP device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `data`: Dat ato write.

int **icsp_soft_make_transition**(**struct** *icsp_soft_driver_t* *self_p, int *transition*)

Make given transition in the ICSP TAP controller state machine.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `transition`: Transition to take in the state machine (one(1) or zero(0)).

**struct icsp_soft_driver_t**

### Public Members

**struct** pin_device_t ***pgec_p**

**struct** pin_device_t ***pged_p**

**struct** pin_device_t ***mclrn_p**

## jtag_soft — Software JTAG

---

Source code: src/drivers/network/jtag_soft.h, src/drivers/network/jtag_soft.c

Test code: tst/drivers/software/network/jtag_soft/master/main.c

---

### Functions

int **jtag_soft_module_init**(void)

Initialize JTAG soft module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **jtag_soft_init**(**struct** *jtag_soft_driver_t* **self_p*, **struct** pin_device_t **tck_p*, **struct** pin_device_t **tms_p*, **struct** pin_device_t **tdi_p*, **struct** pin_device_t **tdo_p*)

Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `tck_p`: Test clock (TCK) pin device.
- `tms_p`: Test mode select (TMS) pin device.
- `tdi_p`: Test data input (TDI) pin device (our out).
- `tdo_p`: Test data output (TDO) pin device (our in).

int **jtag_soft_start**(**struct** *jtag_soft_driver_t* **self_p*)

Start given JTAG soft driver. Configures all pins and enters the idle state in the JTAG TAP controller state machine.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **jtag_soft_stop**(**struct** *jtag_soft_driver_t *self_p*)
Stop given JTAG soft driver. Enters the reset state in the JTAG TAP controller state machine and then configures all pins as inputs.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **jtag_soft_reset**(**struct** *jtag_soft_driver_t *self_p*)
Reset the JTAG device and then enter the idle state.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **jtag_soft_instruction_write**(**struct** *jtag_soft_driver_t *self_p*, **const** void *buf_p, size_t *number_of_bits*)
Write given instruction to given JTAG device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Instruction to write.
- `size`: Number of bits to write.

int **jtag_soft_data_transfer**(**struct** *jtag_soft_driver_t *self_p*, void *rxbuf_p, **const** void *txbuf_p, size_t *number_of_bits*)
Simultaniuos data read/write operation from/to given JTAG device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `rxbuf_p`: Buffer to read into.
- `txbuf_p`: Buffer to write.
- `size`: Number of bits to transfer.

int **jtag_soft_data_read**(**struct** *jtag_soft_driver_t *self_p*, void *buf_p, size_t *number_of_bits*)
Read data from given JTAG device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bits to read.

int **jtag_soft_data_write**(**struct** *jtag_soft_driver_t* *self_p, **const** void *buf_p, size_t *number_of_bits*)

Write given data to the JTAG device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to write.
- `size`: Number of bits to write.

int **jtag_soft_make_transition**(**struct** *jtag_soft_driver_t* *self_p, int *transition*)

Make given transition in the JTAG TAP controller state machine.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `transition`: Transition to take in the state machine (one(1) or zero(0)).

**struct jtag_soft_driver_t**

### Public Members

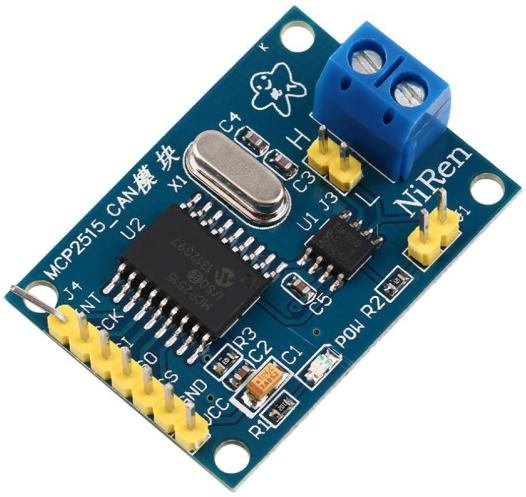
**struct** pin_driver_t **tck**

**struct** pin_driver_t **tms**

**struct** pin_driver_t **tdi**

**struct** pin_driver_t **tdo**

## mcp2515 — CAN BUS chipset




MCP2515 is a CAN controller. Source code: src/drivers/network/mcp2515.h, src/drivers/network/mcp2515.c

Test code: tst/drivers/hardware/network/mcp2515/main.c

---

### Defines

**MCP2515_SPEED_1000KBPS**

**MCP2515_SPEED_500KBPS**

**MCP2515_MODE_NORMAL**

**MCP2515_MODE_LOOPBACK**

### Functions

int **mcp2515_init**(**struct** *mcp2515_driver_t* *self_p, **struct** spi_device_t *spi_p, **struct** pin_device_t *cs_p, **struct** exti_device_t *exti_p, void *chin_p, int *mode*, int *speed*)

Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `spi_p`: SPI driver to use.
- `cs_p`: SPI chip select pin.
- `exti_p`: External interrupt tp use.

- `chin_p`: Frames received from the hardware are written to this channel.
- `mode`: Device mode.
- `speed`: CAN bus speed in kbps.

int **mcp2515_start**(**struct** *mcp2515_driver_t *self_p*)
Starts the CAN device using given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **mcp2515_stop**(**struct** *mcp2515_driver_t *self_p*)
Stops the CAN device referenced by driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

ssize_t **mcp2515_read**(**struct** *mcp2515_driver_t *self_p*, **struct** *mcp2515_frame_t *frame_p*)
Read a CAN frame.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `frame_p`: Read frame.

ssize_t **mcp2515_write**(**struct** *mcp2515_driver_t *self_p*, **const** **struct** *mcp2515_frame_t *frame_p*)
Write a CAN frame.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `frame_p`: Frame to write.

**struct mcp2515_frame_t**

### Public Members

uint32_t **id**

int **size**

int **rtr**

uint32_t **timestamp**

uint8_t **data**[8]

**struct mcp2515_driver_t**

### Public Functions

**mcp2515_driver_t::THRD_STACK(stack, 1024)**

### Public Members

**struct** spi_driver_t **spi**

**struct** exti_driver_t **exti**

int **mode**

int **speed**

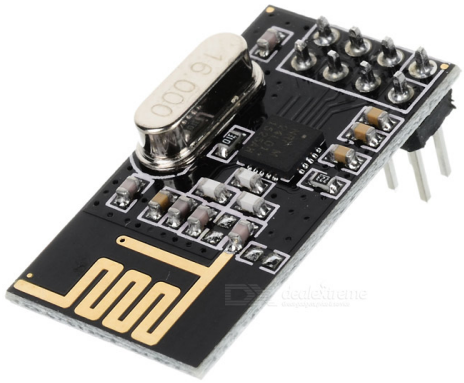
**struct** *chan_t* **chout**

**struct** *chan_t* ***chin_p**

**struct** *sem_t* **isr_sem**

**struct** *sem_t* **tx_sem**

## nrf24l01 — Wireless communication



NRF24L01 is a wireless communication module. Source code: src/drivers/network/nrf24l01.h, src/drivers/network/nrf24l01.c

---

### Functions

int **nrf24l01_module_init**(void)

Initialize NRF24L01 module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **nrf24l01_init**(**struct** *nrf24l01_driver_t* *self_p, **struct** spi_device_t *spi_p, **struct** pin_device_t *cs_p, **struct** pin_device_t *ce_p, **struct** exti_device_t *exti_p, uint32_t *address*)

Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `spi_p`: SPI device.
- `cs_p`: Chip select pin device.
- `ce_p`: CE pin device.
- `exti_p`: External interrupt flagdevice.
- `address`: 4 MSB:s of RX pipes. LSB is set to 0 through 5 for the 6 pipes.

int **nrf24l01_start**(**struct** *nrf24l01_driver_t* *self_p)

Starts the NRF24L01 device using given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **nrf24l01_stop**(**struct** *nrf24l01_driver_t* *self_p)

Stops the NRF24L01 device referenced by driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

ssize_t **nrf24l01_read**(**struct** *nrf24l01_driver_t* *self_p, void *buf_p, size_t *size*)

Read data from the NRF24L01 device.

**Return** Number of received bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read (must be 32).

ssize_t **nrf24l01_write**(**struct** *nrf24l01_driver_t* *self_p, uint32_t *address*, uint8_t *pipe*, **const** void *buf_p, size_t *size*)

Write data to the NRF24L01 device.

**Return** number of sent bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `address`: 4 MSB:s of TX address.

- `pipe`: LSB of TX address.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write (must be 32).

**struct nrf24l01_driver_t**

#### Public Members

**struct** spi_driver_t **spi**

**struct** exti_driver_t **exti**

**struct** pin_driver_t **ce**

**struct** *queue_t* **irqchan**

**struct** *queue_t* **chin**

**struct** *thrd_t* ***thrd_p**

uint32_t **address**

char **irqbuf**[8]

char **chinbuf**[32]

char **stack**[256]

### `owi` — One-Wire Interface

Source code: src/drivers/network/owi.h, src/drivers/network/owi.c

Test code: tst/drivers/hardware/network/owi/main.c

---

#### Defines

**OWI_SEARCH_ROM**

**OWI_READ_ROM**

**OWI_MATCH_ROM**

**OWI_SKIP_ROM**

**OWI_ALARM_SEARCH**

#### Functions

int **owi_init**(**struct** *owi_driver_t* **self_p*, **struct** pin_device_t **dev_p*, **struct** *owi_device_t* **devices_p*, size_t *nmemb*)

Initialize driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Pin device to use.
- `devices_p`: Storage for devices found when searching.
- `nmemb`: Number of members in devices.

int **owi_reset**(**struct** *owi_driver_t* *self_p*)
Send reset on one wire bus.

**Return** true(1) if one or more devices are connected to the bus, false(0) if no devices were found, otherwise negative error code.

**Parameters**

- `self_p`: Driver object.

int **owi_search**(**struct** *owi_driver_t* *self_p*)
Search for devices on given one wire bus. The device id of all found devices are stored in the devices array passed to `owi_init()`.

**Return** Number of devices found or negative error code.

**Parameters**

- `self_p`: Driver object.

ssize_t **owi_read**(**struct** *owi_driver_t* *self_p*, void *buf_p*, size_t *size*)
Read into buffer from one wire bus.

**Return** Number of bits read or negative error code.

**Parameters**

- `self_p`: Driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bits to read.

ssize_t **owi_write**(**struct** *owi_driver_t* *self_p*, **const** void *buf_p*, size_t *size*)
Write buffer to given one wire bus.

**Return** Number of bits written or negative error code.

**Parameters**

- `self_p`: Driver object.
- `buf_p`: Buffer to write.
- `size`: Number of bits to write.

**struct owi_device_t**

### Public Members

uint8_t **id**[8]

**struct owi_driver_t**

### Public Members

**struct** pin_driver_t **pin**

**struct** *owi_device_t* ***devices_p**

size_t **nmemb**

size_t **len**

## spi — Serial Peripheral Interface

Source code: src/drivers/network/spi.h, src/drivers/network/spi.c

---

### Defines

**SPI_MODE_SLAVE**

**SPI_MODE_MASTER**

**SPI_SPEED_8MBPS**

**SPI_SPEED_4MBPS**

**SPI_SPEED_2MBPS**

**SPI_SPEED_1MBPS**

**SPI_SPEED_500KBPS**

**SPI_SPEED_250KBPS**

**SPI_SPEED_125KBPS**

### Functions

int **spi_module_init**(void)

Initialize SPI module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **spi_init**(**struct** spi_driver_t *self_p, **struct** spi_device_t *dev_p, **struct** pin_device_t *ss_pin_p, int *mode*, int *speed*, int *polarity*, int *phase*)

Initialize driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `dev_p`: Device to use.
- `ss_pin_p`: Slave select pin device.
- `mode`: Master or slave mode.

- `speed`: Speed in kbps.
- `polarity`: Set to 0 or 1.
- `phase`: Set to 0 or 1.

int **spi_start**(**struct** spi_driver_t *self_p)
Start given SPI driver. Configures the SPI hardware.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **spi_stop**(**struct** spi_driver_t *self_p)
Stop given SPI driver. Deconfigures the SPI hardware if given driver currently ownes the bus.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **spi_take_bus**(**struct** spi_driver_t *self_p)
In multi master application the driver must take ownership of the SPI bus before performing data transfers. Will re-configure the SPI hardware if configured by another driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **spi_give_bus**(**struct** spi_driver_t *self_p)
In multi master application the driver must give ownership of the SPI bus to let other masters take it.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **spi_select**(**struct** spi_driver_t *self_p)
Select the slave by asserting the slave select pin.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **spi_deselect**(**struct** spi_driver_t *self_p)
Deselect the slave by de-asserting the slave select pin.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

ssize_t **spi_transfer**(**struct** spi_driver_t *self_p, void *rxbuf_p, **const** void *txbuf_p, size_t size)
Simultaniuos read/write operation over the SPI bus.

**Return** Number of transferred bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `rxbuf_p`: Buffer to read into.
- `txbuf_p`: Buffer to write.
- `size`: Number of bytes to transfer.

ssize_t **spi_read**(**struct** spi_driver_t *self_p, void *buf_p, size_t size)
Read data from the SPI bus.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to receive.

ssize_t **spi_write**(**struct** spi_driver_t *self_p, **const** void *buf_p, size_t size)
Write data to the SPI bus.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

ssize_t **spi_get**(**struct** spi_driver_t *self_p, uint8_t *data_p)
Get one byte of data from the SPI bus.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `data_p`: Read data.

ssize_t **spi_put**(**struct** spi_driver_t *self_p, uint8_t data)
Put one byte of data to the SPI bus.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `data`: data to write.

### Variables

**struct** spi_device_t **spi_device**[**SPI_DEVICE_MAX**]

### uart — Universal Asynchronous Receiver/Transmitter

Source code: src/drivers/network/uart.h, src/drivers/network/uart.c

Test code: tst/drivers/hardware/network/uart/main.c

---

### Defines

**UART_FRAME_FORMAT_5N1**

**UART_FRAME_FORMAT_5N15**

**UART_FRAME_FORMAT_5N2**

**UART_FRAME_FORMAT_5O1**

**UART_FRAME_FORMAT_5O15**

**UART_FRAME_FORMAT_5O2**

**UART_FRAME_FORMAT_5E1**

**UART_FRAME_FORMAT_5E15**

**UART_FRAME_FORMAT_5E2**

**UART_FRAME_FORMAT_6N1**

**UART_FRAME_FORMAT_6N15**

**UART_FRAME_FORMAT_6N2**

**UART_FRAME_FORMAT_6O1**

**UART_FRAME_FORMAT_6O15**

**UART_FRAME_FORMAT_6O2**

**UART_FRAME_FORMAT_6E1**

**UART_FRAME_FORMAT_6E15**

**UART_FRAME_FORMAT_6E2**

**UART_FRAME_FORMAT_7N1**

**UART_FRAME_FORMAT_7N15**

**UART_FRAME_FORMAT_7N2**

**UART_FRAME_FORMAT_7O1**

**UART_FRAME_FORMAT_7O15**

**UART_FRAME_FORMAT_7O2**

**UART_FRAME_FORMAT_7E1**

**UART_FRAME_FORMAT_7E15**

**UART_FRAME_FORMAT_7E2**

**UART_FRAME_FORMAT_8N1**

**UART_FRAME_FORMAT_8N15**

**UART_FRAME_FORMAT_8N2**

**UART_FRAME_FORMAT_8O1**

**UART_FRAME_FORMAT_8O15**

**UART_FRAME_FORMAT_8O2**

**UART_FRAME_FORMAT_8E1**

**UART_FRAME_FORMAT_8E15**

**UART_FRAME_FORMAT_8E2**

**UART_FRAME_FORMAT_9N1**

**UART_FRAME_FORMAT_9N15**

**UART_FRAME_FORMAT_9N2**

**UART_FRAME_FORMAT_9O1**

**UART_FRAME_FORMAT_9O15**

**UART_FRAME_FORMAT_9O2**

**UART_FRAME_FORMAT_9E1**

**UART_FRAME_FORMAT_9E15**

**UART_FRAME_FORMAT_9E2**

**UART_FRAME_FORMAT_DEFAULT**

**uart_read**(self_p, buf_p, size)
Read data from the UART.

**Return** Number of received bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to receive.

**uart_write**(self_p, buf_p, size)
Write data to the UART.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

### Typedefs

**typedef** int (***uart_rx_filter_cb_t**) (char c)

### Functions

int **uart_module_init** (void)

Initialize UART module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **uart_init** (**struct** uart_driver_t *self_p, **struct** uart_device_t *dev_p, long baudrate, void *rxbuf_p, size_t size)

Initialize driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: Device to use.
- `baudrate`: Baudrate.
- `rxbuf_p`: Reception buffer.
- `size`: Reception buffer size.

int **uart_set_rx_filter_cb** (**struct** uart_driver_t *self_p, *uart_rx_filter_cb_t* rx_filter_cb)

Set the reception filter callback function.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `rx_filter_cb`: Callback to set.

int **uart_start** (**struct** uart_driver_t *self_p)

Starts the UART device using given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **uart_stop** (**struct** uart_driver_t *self_p)

Stops the UART device referenced by driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **uart_set_frame_format**(**struct** uart_driver_t *self_p, int format)

Set the UART frame format (number of data bits, parity and number of stop bits) in the driver object. This function must be called after uart_init(), and before uart_start().

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `format`: An UART_FRAME_FORMAT_* constant.

int **uart_device_start**(**struct** uart_device_t *dev_p, long baudrate)

Starts the UART device using given configration. The UART device group of functions does not use interrupts, but instead polls the hardware for events. The driver and device functions may not be used for the same UART device.

**Return** zero(0) or negative error code.

**Parameters**

- `dev_p`: UART device to start.
- `baudrate`: Baudrate.

int **uart_device_stop**(**struct** uart_device_t *dev_p)

Stops given UART device.

**Return** zero(0) or negative error code.

**Parameters**

- `dev_p`: UART device to stop.

ssize_t **uart_device_read**(**struct** uart_device_t *dev_p, void *buf_p, size_t size)

Read data from the UART. This function does not wait for interrupts, but instead busy-waits for data by polling UART registers.

**Return** Number of received bytes or negative error code.

**Parameters**

- `dev_p`: UART device to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to receive.

ssize_t **uart_device_write**(**struct** uart_device_t *dev_p, **const** void *buf_p, size_t size)

Write data to the UART. This function does not wait for interrupts, but instead busy-waits for data by polling UART registers.

**Return** Number of written bytes or negative error code.

**Parameters**

- `dev_p`: UART device to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

### Variables

**struct** uart_device_t **uart_device**[**UART_DEVICE_MAX**]

## uart_soft — Software Universal Asynchronous Receiver/Transmitter

Source code: src/drivers/network/uart_soft.h, src/drivers/network/uart_soft.c

---

### Defines

**uart_soft_read**(self_p, buf_p, size)
Read data from the UART.

**Return** Number of received bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to receive.

**uart_soft_write**(self_p, buf_p, size)
Write data to the UART.

**Return** number of sent bytes or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

### Functions

int **uart_soft_init**(**struct** *uart_soft_driver_t* *self_p, **struct** pin_device_t *tx_dev_p, **struct** pin_device_t *rx_dev_p, **struct** exti_device_t *rx_exti_dev_p, int baudrate, void *rxbuf_p, size_t size)
Initialize driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `tx_dev_p`: TX pin device.
- `rx_dev_p`: RX pin device.
- `rx_exti_dev_p`: RX pin external interrupt device.
- `baudrate`: Baudrate.

- `rxbuf_p`: Reception buffer.
- `size`: Reception buffer size.

**struct uart_soft_driver_t**

#### Public Members

**struct** pin_driver_t **tx_pin**

**struct** pin_driver_t **rx_pin**

**struct** exti_driver_t **rx_exti**

**struct** *chan_t* **chout**

**struct** *queue_t* **chin**

int **sample_time**

int **baudrate**

### usb — Universal Serial Bus

Source code: src/drivers/network/usb.h, src/drivers/network/usb.c

---

#### Defines

**REQUEST_TYPE_DATA_MASK**

**REQUEST_TYPE_DATA_DIRECTION_HOST_TO_DEVICE**

**REQUEST_TYPE_DATA_DIRECTION_DEVICE_TO_HOST**

**REQUEST_TYPE_TYPE_MASK**

**REQUEST_TYPE_TYPE_STANDARD**

**REQUEST_TYPE_TYPE_CLASS**

**REQUEST_TYPE_TYPE_VENDOR**

**REQUEST_TYPE_RECIPIENT_MASK**

**REQUEST_TYPE_RECIPIENT_DEVICE**

**REQUEST_TYPE_RECIPIENT_INTERFACE**

**REQUEST_TYPE_RECIPIENT_ENDPOINT**

**REQUEST_TYPE_RECIPIENT_OTHER**

**REQUEST_GET_STATUS**

**REQUEST_SET_ADDRESS**

**REQUEST_GET_DESCRIPTOR**

**REQUEST_SET_CONFIGURATION**

**DESCRIPTOR_TYPE_DEVICE**

**DESCRIPTOR_TYPE_CONFIGURATION**

**DESCRIPTOR_TYPE_STRING**

**DESCRIPTOR_TYPE_INTERFACE**

**DESCRIPTOR_TYPE_ENDPOINT**

**DESCRIPTOR_TYPE_INTERFACE_ASSOCIATION**

**DESCRIPTOR_TYPE_RPIPE**

**DESCRIPTOR_TYPE_CDC**

**USB_CLASS_USE_INTERFACE**

**USB_CLASS_AUDIO**

**USB_CLASS_CDC_CONTROL**

**USB_CLASS_HID**

**USB_CLASS_PHYSICAL**

**USB_CLASS_IMAGE**

**USB_CLASS_PRINTER**

**USB_CLASS_MASS_STORAGE**

**USB_CLASS_HUB**

**USB_CLASS_CDC_DATA**

**USB_CLASS_SMART_CARD**

**USB_CLASS_CONTENT_SECURITY**

**USB_CLASS_VIDEO**

**USB_CLASS_PERSONAL_HEALTHCARE**

**USB_CLASS_AUDIO_VIDEO_DEVICES**

**USB_CLASS_BILLBOARD_DEVICE_CLASS**

**USB_CLASS_DIAGNOSTIC_DEVICE**

**USB_CLASS_WIRELESS_CONTROLLER**

**USB_CLASS_MISCELLANEOUS**

**USB_CLASS_APPLICATION_SPECIFIC**

**USB_CLASS_VENDOR_SPECIFIC**

**ENDPOINT_ENDPOINT_ADDRESS_DIRECTION**(address)

**ENDPOINT_ENDPOINT_ADDRESS_NUMBER**(address)

**ENDPOINT_ATTRIBUTES_USAGE_TYPE**(attributes)

**ENDPOINT_ATTRIBUTES_SYNCHRONISATION_TYPE**(attributes)

**ENDPOINT_ATTRIBUTES_TRANSFER_TYPE**(attributes)

**ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_CONTROL**

**ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_ISOCHRONOUS**

**ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_BULK**

**ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_INTERRUPT**

**CONFIGURATION_ATTRIBUTES_BUS_POWERED**

**USB_CDC_LINE_CODING**

**USB_CDC_CONTROL_LINE_STATE**

**USB_CDC_SEND_BREAK**

**USB_MESSAGE_TYPE_ADD**

**USB_MESSAGE_TYPE_REMOVE**

### Functions

int **usb_format_descriptors**(void *_out_p_, uint8_t *_buf_p_, size_t _size_)

Format the descriptors and write them to given channel.

**Return** zero(0) or negative error code.

**Parameters**

- `out_p`: Output channel.
- `buf_p`: Pointer to the descriptors to format.
- `size`: Number of bytes in the descriptors buffer.

**struct** *usb_descriptor_configuration_t* ***usb_desc_get_configuration**(uint8_t *_desc_p_, size_t _size_, int _configuration_)

Get the configuration descriptor for given configuration index.

**Return** Configuration or NULL on failure.

**Parameters**

- `buf_p`: Pointer to the descriptors.
- `size`: Number of bytes in the descriptors buffer.
- `configuration`: Configuration to find.

**struct** *usb_descriptor_interface_t* ***usb_desc_get_interface**(uint8_t *_desc_p_, size_t _size_, int _configuration_, int _interface_)

Get the interface descriptor for given configuration and interface index.

**Return** Interface or NULL on failure.

**Parameters**

- `buf_p`: Pointer to the descriptors.
- `size`: Number of bytes in the descriptors buffer.
- `configuration`: Configuration to find.
- `interface`: Interface to find.

**struct** *usb_descriptor_endpoint_t* ***usb_desc_get_endpoint**(uint8_t *_desc_p_, size_t _size_, int _configuration_, int _interface_, int _endpoint_)

Get the endpoint descriptor for given configuration, interface and endpoint index.

**Return** Endpoint or NULL on failure.

**Parameters**

- `buf_p`: Pointer to the descriptors.
- `size`: Number of bytes in the descriptors buffer.
- `configuration`: Configuration to find.
- `interface`: Interface to find.
- `endpoint`: Endpoint to find.

int **usb_desc_get_class**(uint8_t **buf_p*, size_t *size*, int *configuration*, int *interface*)
Get the interface class.

**Return**

**Parameters**

- `buf_p`: Pointer to the descriptors.
- `size`: Number of bytes in the descriptors buffer.
- `configuration`: Configuration to find.
- `interface`: Interface to find.

### Variables

**struct** usb_device_t **usb_device**[**USB_DEVICE_MAX**]

**struct usb_setup_t**

#### Public Members

uint8_t **request_type**

uint8_t **request**

uint16_t **feature_selector**

uint16_t **zero_interface_endpoint**

**struct usb_setup_t::@6::@7 usb_setup_t::clear_feature**

uint16_t **zero0**

uint16_t **zero1**

**struct usb_setup_t::@6::@8 usb_setup_t::get_configuration**

uint8_t **descriptor_index**

uint8_t **descriptor_type**

uint16_t **language_id**

**struct usb_setup_t::@6::@9 usb_setup_t::get_descriptor**

uint16_t **device_address**

uint16_t **zero**

struct usb_setup_t::@6::@10 usb_setup_t::set_address

uint16_t configuration_value

struct usb_setup_t::@6::@11 usb_setup_t::set_configuration

uint16_t value

uint16_t index

struct usb_setup_t::@6::@12 usb_setup_t::base

union usb_setup_t::@6 usb_setup_t::u

uint16_t length

struct usb_descriptor_header_t

#### Public Members

uint8_t length

uint8_t descriptor_type

struct usb_descriptor_device_t

#### Public Members

uint8_t length

uint8_t descriptor_type

uint16_t bcd_usb

uint8_t device_class

uint8_t device_subclass

uint8_t device_protocol

uint8_t max_packet_size_0

uint16_t id_vendor

uint16_t id_product

uint16_t bcd_device

uint8_t manufacturer

uint8_t product

uint8_t serial_number

uint8_t num_configurations

struct usb_descriptor_configuration_t

#### Public Members

uint8_t length

uint8_t descriptor_type

uint16_t **total_length**

uint8_t **num_interfaces**

uint8_t **configuration_value**

uint8_t **configuration**

uint8_t **configuration_attributes**

uint8_t **max_power**

**struct usb_descriptor_interface_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **interface_number**

uint8_t **alternate_setting**

uint8_t **num_endpoints**

uint8_t **interface_class**

uint8_t **interface_subclass**

uint8_t **interface_protocol**

uint8_t **interface**

**struct usb_descriptor_endpoint_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **endpoint_address**

uint8_t **attributes**

uint16_t **max_packet_size**

uint8_t **interval**

**struct usb_descriptor_string_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **string**[256]

**struct usb_descriptor_interface_association_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **first_interface**

uint8_t **interface_count**

uint8_t **function_class**

uint8_t **function_subclass**

uint8_t **function_protocol**

uint8_t **function**

**struct usb_descriptor_cdc_header_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **sub_type**

uint16_t **bcd**

**struct usb_descriptor_cdc_acm_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **sub_type**

uint8_t **capabilities**

**struct usb_descriptor_cdc_union_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **sub_type**

uint8_t **master_interface**

uint8_t **slave_interface**

**struct usb_descriptor_cdc_call_management_t**

#### Public Members

uint8_t **length**

uint8_t **descriptor_type**

uint8_t **sub_type**

uint8_t **capabilities**

uint8_t **data_interface**

**union usb_descriptor_t**

#### Public Members

**struct** *usb_descriptor_header_t* **header**

**struct** *usb_descriptor_device_t* **device**

**struct** *usb_descriptor_configuration_t* **configuration**

**struct** *usb_descriptor_interface_t* **interface**

**struct** *usb_descriptor_endpoint_t* **endpoint**

**struct** *usb_descriptor_string_t* **string**

**struct usb_cdc_line_info_t**

#### Public Members

uint32_t **dte_rate**

uint8_t **char_format**

uint8_t **parity_type**

uint8_t **data_bits**

**struct usb_message_header_t**

#### Public Members

int **type**

**struct usb_message_add_t**

#### Public Members

**struct** *usb_message_header_t* **header**

int **device**

**union usb_message_t**

### Public Members

**struct** *usb_message_header_t* **header**

**struct** *usb_message_add_t* **add**

## usb_device — Universal Serial Bus - Device

A USB device is powered and enumerated by a USB host.

The implementation of this module aims to be simple, but yet flexible. It's possible to change the USB configuration descriptors at runtime by stopping the current driver, initialize a new driver and start the new driver. For simple devices only a single configuration is normally needed.

Using the USB device module is fairly easy. First write the USB descriptors, then initialize the class drivers, then initialize the USB device driver and then start it.

See the test code below for an example usage.

---

Class driver modules:

## usb_device_class_cdc — CDC ACM (serial port over USB)

USB CDC (Communications Device Class) ACM (Abstract Control Model) is a vendor-independent publicly documented protocol that can be used for emulating serial ports over USB.

More information on Wikipedia.

---

Source code: src/drivers/network/usb/device/class/cdc.h, src/drivers/network/usb/device/class/cdc.c

Test code: tst/drivers/hardware/network/usb_device/main.c

---

### Defines

**usb_device_class_cdc_read**(self_p, buf_p, size)
Read data from the CDC driver.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

**usb_device_class_cdc_write**(self_p, buf_p, size)
Write data to the CDC driver.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

### Functions

int **usb_device_class_cdc_module_init**(void)
Initialize the CDC module.

**Return** zero(0) or negative error code.

int **usb_device_class_cdc_init**(**struct** *usb_device_class_cdc_driver_t* **self_p*, int *control_interface*, int *endpoint_in*, int *endpoint_out*, void **rxbuf_p*, size_t *size*)
Initialize driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `rxbuf_p`: Reception buffer.
- `size`: Reception buffer size.

int **usb_device_class_cdc_input_isr**(**struct** *usb_device_class_cdc_driver_t* **self_p*)
Called by the USB device driver periodically to let the CDC driver read received data from the hardware.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **usb_device_class_cdc_is_connected**(**struct** *usb_device_class_cdc_driver_t* **self_p*)
Check if the CDC is connected to the remote endpoint.

**Return** true(1) if connected, false(0) if disconnected, otherwise negative error code.

**Parameters**

- `self_p`: Initialized driver object.

**struct usb_device_class_cdc_driver_t**

#### Public Members

**struct** usb_device_driver_base_t **base**

**struct** usb_device_driver_t ***drv_p**

int **control_interface**

int **endpoint_in**

int **endpoint_out**

int **line_state**

**struct** *usb_cdc_line_info_t* **line_info**

**struct** *chan_t* **chout**

**struct** *queue_t* **chin**

---

Source code: src/drivers/network/usb_device.h, src/drivers/network/usb_device.c

Test code: tst/drivers/hardware/network/usb_device/main.c

---

## Functions

int **usb_device_module_init**(void)

Initialize the USB device module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

**int usb_device_init(struct usb_device_driver_t * self_p, struct usb_device_t * dev_p, s**

Initialize the USB device driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `dev_p`: USB device to use.
- `drivers_pp`: An array of initialized drivers.
- `drivers_max`: Length of the drivers array.
- `descriptors_pp`: A NULL terminated array of USB descriptors.

int **usb_device_start**(**struct** usb_device_driver_t *self_p)

Start the USB device device using given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **usb_device_stop**(**struct** usb_device_driver_t *self_p)

Stop the USB device device referenced by driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

ssize_t **usb_device_write**(**struct** usb_device_driver_t *self_p, int endpoint, **const** void *buf_p, size_t size)

Write data to given endpoint.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `endpoint`: Endpoint to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

ssize_t **usb_device_read_isr**(**struct** usb_device_driver_t *self_p, int endpoint, void *buf_p, size_t size)

Read data from given endpoint from an isr or with the system lock taken.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `endpoint`: Endpoint to read data from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **usb_device_write_isr**(**struct** usb_device_driver_t *self_p, int endpoint, **const** void *buf_p, size_t size)

Write data to given endpoint from an isr or with the system lock taken.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `endpoint`: Endpoint to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

### `usb_host` — Universal Serial Bus - Host

A USB host powers the bus and enumerates connected USB devices.

---

Class driver modules:

### usb_host_class_hid — Human Interface Device (HID)

In computing, the USB human interface device class (USB HID class) is a part of the USB specification for computer peripherals: it specifies a device class (a type of computer hardware) for human interface devices such as keyboards, mice, game controllers and alphanumeric display devices.

More information on Wikipedia.

---

Source code: src/drivers/network/usb/host/class/hid.h, src/drivers/network/usb/host/class/hid.c

---

#### Defines

**USB_CLASS_HID_SUBCLASS_NONE**

**USB_CLASS_HID_SUBCLASS_BOOT_INTERFACE**

**USB_CLASS_HID_PROTOCOL_NONE**

**USB_CLASS_HID_PROTOCOL_KEYBOARD**

**USB_CLASS_HID_PROTOCOL_MOUSE**

#### Functions

int **usb_host_class_hid_init**(**struct** *usb_host_class_hid_driver_t* *self_p, **struct** usb_host_driver_t *usb_p, **struct** *usb_host_class_hid_device_t* *devices_p, size_t *length*)

Initialize driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `usb_p`: USB driver to use.
- `devices_p`: Array of devices. One entry in this array is allocated for each HID device that is connected to the host.
- `length`: Length of the devices array.

int **usb_host_class_hid_start**(**struct** *usb_host_class_hid_driver_t* *self_p)

Starts the HID driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object to start.

int **usb_host_class_hid_stop**(**struct** *usb_host_class_hid_driver_t* *self_p)

Stops the HID driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized to stop.

**struct usb_host_class_hid_device_t**

### Public Members

uint8_t **buf**[1]

**struct usb_host_class_hid_driver_t**

### Public Members

**struct** usb_host_driver_t ***usb_p**

**struct** *usb_host_class_hid_device_t* ***devices_p**

size_t **length**

size_t **size**

**struct usb_host_class_hid_driver_t::@4 usb_host_class_hid_driver_t::report**

**struct** *usb_host_device_driver_t* **device_driver**

## usb_host_class_mass_storage — Mass Storage

The USB mass storage device class (also known as USB MSC or UMS) is a set of computing communications protocols defined by the USB Implementers Forum that makes a USB device accessible to a host computing device and enables file transfers between the host and the USB device. To a host, the USB device acts as an external hard drive; the protocol set interfaces with a number of storage devices.

More information on Wikipedia.

---

Source code: src/drivers/network/usb/host/class/mass_storage.h, src/drivers/network/usb/host/class/mass_storage.c

---

### Functions

int **usb_host_class_mass_storage_init**(**struct** *usb_host_class_mass_storage_driver_t* *self_p, **struct** usb_host_driver_t *usb_p, **struct** *usb_host_class_mass_storage_device_t* *devices_p, size_t length)

int **usb_host_class_mass_storage_start**(**struct** *usb_host_class_mass_storage_driver_t* *self_p)

int **usb_host_class_mass_storage_stop**(**struct** *usb_host_class_mass_storage_driver_t* *self_p)

ssize_t **usb_host_class_mass_storage_device_read**(**struct** *usb_host_device_t* *device_p, void *buf_p, size_t address, size_t size)

**struct usb_host_class_mass_storage_device_t**

### Public Members

uint8_t **buf**[1]

**struct usb_host_class_mass_storage_driver_t**

### Public Members

**struct** usb_host_driver_t ***usb_p**

**struct** *usb_host_class_mass_storage_device_t* ***devices_p**

size_t **length**

size_t **size**

**struct usb_host_class_mass_storage_driver_t::@5 usb_host_class_mass_storage_driver_t:**

**struct** *usb_host_device_driver_t* **device_driver**

---

Source code: src/drivers/network/usb_host.h, src/drivers/network/usb_host.c

---

## Defines

**USB_HOST_DEVICE_STATE_NONE**

**USB_HOST_DEVICE_STATE_ATTACHED**

**USB_PIPE_TYPE_CONTROL**

**USB_PIPE_TYPE_INTERRUPT**

**USB_PIPE_TYPE_ISOCHRONOUS**

**USB_PIPE_TYPE_BULK**

## Functions

int **usb_host_module_init**(void)

Initialize the USB host module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **usb_host_init**(**struct** usb_host_driver_t *self_p, **struct** usb_device_t *dev_p, **struct** *usb_host_device_t* *devices_p, size_t length)

Initialize the USB host driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.

- `dev_p`: USB device to use.
- `devices_p`: An array of devices. One entry in this array is allocated for each USB device that is connected to the host.
- `length`: Length of the devices array.

int **usb_host_start**(**struct** usb_host_driver_t *self_p)

Start the USB host device using given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **usb_host_stop**(**struct** usb_host_driver_t *self_p)

Stop the USB host device referenced by driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **usb_host_driver_add**(**struct** usb_host_driver_t *self_p, **struct** *usb_host_device_driver_t* *driver_p, void *arg_p)

Add given class/vendor driver to the USB host driver.

When a USB device is plugged in, its class and vendor information is read by the host. Those values are used to find the device driver for this particular device. If there is no driver, the device cannot be configured and will not work.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `driver_p`: USB device driver to add.

int **usb_host_driver_remove**(**struct** usb_host_driver_t *self_p, **struct** *usb_host_device_driver_t* *driver_p)

Remove given class/vendor driver from the USB host driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `driver_p`: USB device driver to remove.

**struct** *usb_host_device_t* ***usb_host_device_open**(**struct** usb_host_driver_t *self_p, int *device*)

Open given device in given driver. Open a device before reading and writing data to it with usb_host_device_read() or usb_host_device_write().

**Return** Opened device or NULL on failure.

**Parameters**

- `self_p`: Initialized driver.

- `device`: Device to open.

int **usb_host_device_close**(**struct** usb_host_driver_t *self_p, int *device*)
Close given device in given driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver.
- `device`: Device to close.

ssize_t **usb_host_device_read**(**struct** *usb_host_device_t* *device_p, int *endpoint*, void *buf_p, size_t *size*)
Read data from given endpoint for given device.

**Return** Number of bytes read or negative error code.

**Parameters**

- `device_p`: Device to read from.
- `endpoint`: Endpoint to read data from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **usb_host_device_write**(**struct** *usb_host_device_t* *device_p, int *endpoint*, **const** void *buf_p, size_t *size*)
Write data to given endpoint for given device.

**Return** Number of bytes written or negative error code.

**Parameters**

- `device_p`: Device to write to.
- `endpoint`: Endpoint to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

ssize_t **usb_host_device_control_transfer**(**struct** *usb_host_device_t* *device_p, **struct** *usb_setup_t* *setup_p, void *buf_p, size_t *size*)
Perform a control transfer on endpoint zero(0).

A control transfer can have up to three stages. First the setup stage, then an optional data stage, and at last a status stage.

**Return** Number of bytes read/written or negative error code.

**Parameters**

- `device_p`: Device to write to.
- `setup_p`: Setup packet to write.
- `buf_p`: Buffer to read/write. May be NULL if no data shall be transferred.
- `size`: Number of bytes to read/write.

int **usb_host_device_set_configuration**(**struct** *usb_host_device_t* *device_p*, uint8_t *configuration*)

Set configuration for given device.

**Return** zero(0) or negative error code.

**Parameters**

- `device_p`: Device to use.
- `configuration`: Configuration to set.

**struct usb_host_device_t**

*#include <usb_host.h>* An USB device as seen by the host.

### Public Members

int **id**

int **state**

int **address**

int **vid**

int **pid**

char ***description_p**

size_t **max_packet_size**

uint8_t **configuration**

**struct** *usb_descriptor_device_t* ***dev_p**

**struct** *usb_descriptor_configuration_t* ***conf_p**

**struct usb_host_device_t::@13::@15 usb_host_device_t::descriptor**

**struct usb_host_device_t::@13 usb_host_device_t::current**

**struct** usb_host_driver_t ***self_p**

**struct** usb_pipe_t ***pipes**[32]

size_t **size**

uint8_t **buf**[128]

**struct usb_host_device_t::@14 usb_host_device_t::descriptors**

**struct usb_host_device_driver_t**

*#include <usb_host.h>* Used to find a device driver.

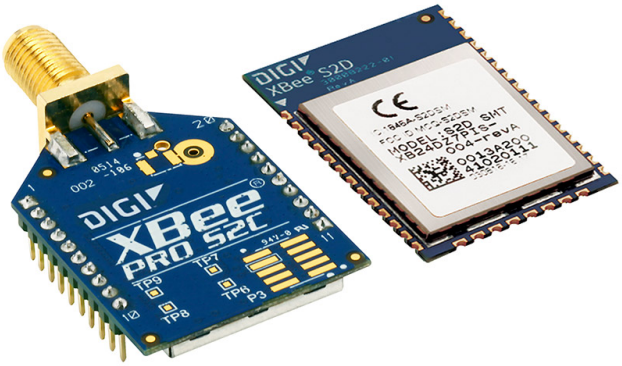
### Public Members

int (***supports**)(**struct** *usb_host_device_t* *)

int (***enumerate**)(**struct** *usb_host_device_t* *)

**struct** *usb_host_device_driver_t* ***next_p**

## xbee — XBee




An XBee is a module for wireless communication. This driver implements reception and transmission of frames over a channel (normally a UART driver).

Known limitations:

- Only AP=2 is supported. That is, API operation with escaped charaters. AP=1 could easily be implemented.

Manufacturer homepage: https://www.digi.com/

---

Source code: src/drivers/network/xbee.h, src/drivers/network/xbee.c

Test code: tst/drivers/software/network/xbee/main.c

Test coverage: src/drivers/network/xbee.c

Example code: examples/xbee/main.c

---

### Defines

**XBEE_DATA_MAX**

**XBEE_FRAME_ID_NO_ACK**

**XBEE_FRAME_TYPE_TX_REQUEST_64_BIT_ADDRESS**

**XBEE_FRAME_TYPE_TX_REQUEST_16_BIT_ADDRESS**

**XBEE_FRAME_TYPE_AT_COMMAND**

**XBEE_FRAME_TYPE_AT_COMMAND_QUEUE_PARAMETER_VALUE**

**XBEE_FRAME_TYPE_ZIGBEE_TRANSMIT_REQUEST**

**XBEE_FRAME_TYPE_EXPLICIT_ADDRESSING_ZIGBEE_COMMAND_FRAME**

**XBEE_FRAME_TYPE_REMOTE_COMMAND_REQUEST**

**XBEE_FRAME_TYPE_CREATE_SOURCE_ROUTE**

**XBEE_FRAME_TYPE_RX_PACKET_64_BIT_ADDRESS**

**XBEE_FRAME_TYPE_RX_PACKET_16_BIT_ADDRESS**

**XBEE_FRAME_TYPE_RX_PACKET_64_BIT_ADDRESS_IO**

**XBEE_FRAME_TYPE_RX_PACKET_16_BIT_ADDRESS_IO**

**XBEE_FRAME_TYPE_AT_COMMAND_RESPONSE**

**XBEE_FRAME_TYPE_TX_STATUS**

**XBEE_FRAME_TYPE_MODEM_STATUS**

**XBEE_FRAME_TYPE_ZIGBEE_TRANSMIT_STATUS**

**XBEE_FRAME_TYPE_ZIGBEE_RECEIVE_PACKET_AO_0**

**XBEE_FRAME_TYPE_ZIGBEE_EXPLICIT_RX_INDICATOR_AO_1**

**XBEE_FRAME_TYPE_ZIGBEE_IO_DATA_SAMPLE_RX_INDICATOR**

**XBEE_FRAME_TYPE_XBEE_SENSOR_READ_INDICATOR_AO_0**

**XBEE_FRAME_TYPE_NODE_IDENTIFICATION_INDICATOR_AO_0**

**XBEE_FRAME_TYPE_REMOTE_COMMAND_RESPONSE**

**XBEE_FRAME_TYPE_EXTENDED_MODEM_STATUS**

**XBEE_FRAME_TYPE_OVER_THE_AIR_FIRMWARE_UPDATE_STATUS**

**XBEE_FRAME_TYPE_ROUTE_RECORD_INDICATOR**

**XBEE_FRAME_TYPE_MANY_TO_ONE_ROUTE_REQUEST_INDICATOR**

**XBEE_PIN_DIO0**

**XBEE_PIN_DIO1**

**XBEE_PIN_DIO2**

**XBEE_PIN_DIO3**

**XBEE_PIN_DIO4**

**XBEE_PIN_DIO5**

**XBEE_PIN_DIO6**

**XBEE_PIN_DIO7**

**XBEE_PIN_DIO8**

**XBEE_PIN_AD0**

**XBEE_PIN_AD1**

**XBEE_PIN_AD2**

**XBEE_PIN_AD3**

**XBEE_PIN_MODE_DISABLED**

**XBEE_PIN_MODE_ADC**

**XBEE_PIN_MODE_INPUT**

**XBEE_PIN_MODE_OUTPUT_LOW**

**XBEE_PIN_MODE_OUTPUT_HIGH**

### Functions

int **xbee_module_init**(void)
Initialize the xbee module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **xbee_init**(**struct** *xbee_driver_t* *self_p, void *chin_p, void *chout_p)
Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `chin_p`: Input channel from the XBee module, often a UART driver.
- `chout_p`: Output channel to the XBee module, often a UART driver output channel.

int **xbee_read**(**struct** *xbee_driver_t* *self_p, **struct** *xbee_frame_t* *frame_p)
Read a frame from the XBee module. Blocks until a frame is received or an error occurs.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `frame_p`: Frame to read into.

int **xbee_write**(**struct** *xbee_driver_t* *self_p, **const struct** *xbee_frame_t* *frame_p)
Write given frame to the XBee module. Blocks until the frame have been transmitted or an error occurs.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `frame_p`: Frame to write.

int **xbee_print_frame**(void *chan_p, **struct** *xbee_frame_t* *frame_p)
Decode given frame and write it as a human readable string to given channel.

**Return** zero(0) or negative error code.

**Parameters**

- `chan_p`: Channel to write the human readable string to.
- `frame_p`: Frame to decode.

**const** char ***xbee_frame_type_as_string**(uint8_t *frame_type*)
Map given frame type to a human readable string.

**Return** Human readable frame type string or NULL if given frame type is unknown.

**Parameters**

- `frame_type`: Frame type.

**const** char ***xbee_tx_status_as_string**(uint8_t *tx_status*)
Map given TX status to a human readable string.

**Return** Human readable TX status string or NULL if given TX status is unknown.

**Parameters**

- `tx_status`: TX status.

**const** char ***xbee_modem_status_as_string**(uint8_t *modem_status*)
Map given modem status to a human readable string.

**Return** Human readable modem status string or NULL if given modem status is unknown.

**Parameters**

- `modem_status`: Modem status.

**const** char ***xbee_at_command_response_status_as_string**(uint8_t *response_status*)
Map given AT command response status to a human readable string.

**Return** Human readable AT command response status string or NULL if given response status is unknown.

**Parameters**

- `response_status`: AT command response status.

**struct xbee_frame_t**

### Public Members

uint8_t **type**

uint8_t **buf**[**XBEE_DATA_MAX**]

size_t **size**

**struct xbee_frame_t::@16 xbee_frame_t::data**

**struct xbee_driver_t**

### Public Members

**struct** *chan_t* ***chin_p**

**struct** *chan_t* ***chout_p**

**struct xbee_driver_t::@17 xbee_driver_t::transport**

### xbee_client — XBee client

This is a high level driver for XBee modules, providing functions to execute AT commands, and receive and transmit data from/to a remote XBee module.

An XBee client consists of an object and a thread. The thread asynchronously receives frames from the XBee module, leaving this logic out of the user application. The user can read received packets (stored in a queue in the client object), transmit data with or without acknowledgement, and execute AT commands with acknowledgement.

### Example usage

This example shows how to receive and tramsmit data from/to a remote XBee module and control IO on the local XBee module.

See examples/xbee_client/main.c for the full example, including initialization and error handling.

Declaration of the variables used in this example:

```
static struct xbee_client_t client;
ssize_t size;
uint8_t buf[XBEE_DATA_MAX];
struct xbee_client_address_t sender;

/* See full example for initialization. */
```

Reception and transmission of packets from/to a remote XBee module:

```
/* Wait for a packet from a remote XBee module. */
size = xbee_client_read_from(&client,
                             &buf[0],
                             sizeof(buf),
                             &sender);

/* Print the read packet and its sender. */
xbee_client_print_address(sys_get_stdout(), &sender);
std_hexdump(sys_get_stdout(), &buf[0], size);

/* Send the packet back to the sender. */
size = xbee_client_write_to(&client,
                            &buf[0],
                            sizeof(buf),
                            0,
                            &sender);
```

The classic blink example using a LED connected to the XBee:

```
/* Configure DIO0 as output. */
xbee_client_pin_set_mode(&client,
                         XBEE_PIN_DIO0,
                         XBEE_CLIENT_PIN_OUTPUT);

/* Blink the LED. */
while (1) {
   xbee_client_pin_toggle(&client, XBEE_PIN_DIO0);
   thrd_sleep(0.5);
}
```

Source code: src/drivers/network/xbee_client.h, src/drivers/network/xbee_client.c

Example code: examples/xbee_client/main.c

### Defines

**XBEE_CLIENT_PIN_OUTPUT**

**XBEE_CLIENT_PIN_INPUT**

**XBEE_CLIENT_PIN_ADC**

**XBEE_CLIENT_NON_BLOCKING_READ**

**XBEE_CLIENT_NO_ACK**

### Enums

**enum xbee_client_address_type_t**

*Values:*

**xbee_client_address_type_invalid_t** = 0

**xbee_client_address_type_16_bits_t**

**xbee_client_address_type_64_bits_t**

### Functions

int **xbee_client_module_init**(void)

Initialize the xbee module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **xbee_client_init**(**struct** *xbee_client_t* *self_p, void *chin_p, void *chout_p, void *buf_p, size_t *size*, int *flags*)

Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `chin_p`: Input channel from the XBee module, often a UART driver.
- `chout_p`: Output channel to the XBee module, often a UART driver output channel.
- `buf_p`: Frame reception buffer.
- `size`: Frame reception buffer size in words.

- `flags`: Client configuration flags. May be any combination of `XBEE_CLIENT_NON_BLOCKING_READ`.

void ***xbee_client_main**(void *_arg_p_)
The client thread entry function.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.

ssize_t **xbee_client_read_from**(**struct** *xbee_client_t* *_self_p_, void *_buf_p_, size_t _size_, **struct** *xbee_client_address_t* *_address_p_)
Read data from one RX packet into given buffer. The sender 16 or 64 bits address is written to `address_p`.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.
- `address_p`: Sender address.

ssize_t **xbee_client_write_to**(**struct** *xbee_client_t* *_self_p_, **const** void *_buf_p_, size_t _size_, int _flags_, **struct** *xbee_client_address_t* *_address_p_)
Create a TX packet of given data and write it to given 16 or 64 bits XBee address.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.
- `flags`: Read flags.
- `flags`: May be any combination of `XBEE_CLIENT_NO_ACK`.
- `address_p`: Receiver address.

int **xbee_client_pin_set_mode**(**struct** *xbee_client_t* *_self_p_, int _pin_, int _mode_)
Configure given pin to given mode.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `pin`: Pin to set mode for.
- `mode`: Pin mode to set.

int **xbee_client_pin_read**(**struct** *xbee_client_t* *_self_p_, int _pin_)
Read the current value of given pin.

**Return** zero(0), one(1) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `pin`: Pin to read from.

int **xbee_client_pin_write**(**struct** *xbee_client_t* *self_p, int *pin*, int *value*)
Write given value to given pin.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `pin`: Pin to write to.
- `value`: `1` for high and `0` for low output.

int **xbee_client_pin_toggle**(**struct** *xbee_client_t* *self_p, int *pin*)
Toggle the pin output value (high/low).

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `pin`: Pin to toggle.

int **xbee_client_pin_convert**(**struct** *xbee_client_t* *self_p, int *pin*)
Start a synchronous convertion of an analog signal a digital sample.

**Return** Sample or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `pin`: ADC pin to convert.

ssize_t **xbee_client_at_command_read**(**struct** *xbee_client_t* *self_p, **const** char *command_p, uint8_t *parameter_p, size_t *size*)
Execute given AT command and store its read parameter in given buffer.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.
- `parameter_p`: Parameter buffer to read into.
- `size`: Buffer size in bytes.

int **xbee_client_at_command_write**(**struct** *xbee_client_t* *self_p, **const** char *command_p, **const** uint8_t *parameter_p, size_t *size*)
Execute given AT command with given parameter.

**Return** zero(0 or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.
- `parameter_p`: Parameter buffer to write.
- `size`: Buffer size in bytes.

int **xbee_client_at_command_read_u8**(**struct** *xbee_client_t* *self_p, **const** char *command_p, uint8_t *parameter_p)

Execute given AT command and store its read 8 bits parameter in given variable.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.
- `parameter_p`: Read parameter.

int **xbee_client_at_command_write_u8**(**struct** *xbee_client_t* *self_p, **const** char *command_p, uint8_t *parameter*)

Execute given AT command with given 8 bits parameter.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.
- `parameter`: Parameter to write.

int **xbee_client_at_command_read_u16**(**struct** *xbee_client_t* *self_p, **const** char *command_p, uint16_t *parameter_p)

Execute given AT command and store its read 16 bits parameter in given variable.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.
- `parameter_p`: Read parameter.

int **xbee_client_at_command_write_u16**(**struct** *xbee_client_t* *self_p, **const** char *command_p, uint16_t *parameter*)

Execute given AT command with given 16 bits parameter.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.

- `parameter`: Parameter to write.

int **xbee_client_at_command_read_u32**(**struct** *xbee_client_t* *self_p, **const** char *command_p, uint32_t *parameter_p)

Execute given AT command and store its read 32 bits parameter in given variable.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.
- `parameter_p`: Read parameter.

int **xbee_client_at_command_write_u32**(**struct** *xbee_client_t* *self_p, **const** char *command_p, uint32_t *parameter*)

Execute given AT command with given 32 bits parameter.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized client object.
- `command_p`: Two letters AT command.
- `parameter`: Parameter to write.

int **xbee_client_print_address**(void *chan_p, **struct** *xbee_client_address_t* *address_p)

Format and print given address to given channel.

**Return** zero(0) or negative error code.

**Parameters**

- `chan_p`: Channel to print to.
- `address_p`: Address to print.

**struct xbee_client_address_t**

### Public Members

*xbee_client_address_type_t* **type**

uint8_t **buf**[8]

**struct xbee_client_t**

### Public Members

**struct** *queue_t* **chin**

**struct** *xbee_driver_t* **driver**

uint8_t **frame_id**

uint8_t **value**

**struct xbee_client_t::@18 xbee_client_t::pins**

**struct** *mutex_t* **mutex**

**struct** *cond_t* **cond**

**struct** *xbee_frame_t* ***frame_p**

void ***buf_p**

size_t ***size_p**

int **res**

**struct xbee_client_t::@19::@20 xbee_client_t::rx**

**struct xbee_client_t::@19 xbee_client_t::rpc**

**struct** *log_object_t* **log**

### displays

Display drivers.

#### hd44780 — Dot matrix LCD

Source code: src/drivers/displays/hd44780.h, src/drivers/displays/hd44780.c

Example code: examples/drivers/displays/hd44780/main.c

---

### Functions

int **hd44780_module_init**(void)

Initialize the driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **hd44780_init**(**struct** *hd44780_driver_t* **self_p*, **struct** pin_device_t **rs_p*, **struct** pin_device_t **enable_p*, **struct** pin_device_t **data_4_p*, **struct** pin_device_t **data_5_p*, **struct** pin_device_t **data_6_p*, **struct** pin_device_t **data_7_p*, unsigned int *number_of_rows*, unsigned int *number_of_columns*)

Initialize driver object. The driver object will be used for a single display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `rs_p`: RS pin device.
- `enable_p`: RS pin device.
- `data_4_p`: Data 4 pin device.
- `data_5_p`: Data 5 pin device.

- `data_6_p`: Data 6 pin device.
- `data_7_p`: Data 7 pin device.
- `number_of_rows`: Number of rows.
- `number_of_columns`: Number of columns.

int **hd44780_start**(**struct** *hd44780_driver_t* *self_p)
Start the driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_stop**(**struct** *hd44780_driver_t* *self_p)
Stop the driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_display**(**struct** *hd44780_driver_t* *self_p, **const** char *text_p)
Display given text on the display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `text_p`: Text to display.

int **hd44780_write**(**struct** *hd44780_driver_t* *self_p, **const** char *text_p)
Write given text to the display at current cursor position.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `text_p`: Text to write to the display.

int **hd44780_put**(**struct** *hd44780_driver_t* *self_p, char *character*)
Write given character to the display at current cursor position.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `character`: Character to write to the display.

int **hd44780_clear**(**struct** *hd44780_driver_t* *self_p)
Clear the display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_cursor_move**(**struct** *hd44780_driver_t* *self_p, unsigned int *row*, unsigned int *column*)
Move the cursor to given row and column, relative to the upper left-hand corner of the screen, which is (0, 0).

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `row`: Row to move to.
- `column`: Colum to move to.

int **hd44780_cursor_show**(**struct** *hd44780_driver_t* *self_p)
Show the cursor on the display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_cursor_hide**(**struct** *hd44780_driver_t* *self_p)
Do not show the cursor on the display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_text_show**(**struct** *hd44780_driver_t* *self_p)
Show the text on the display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_text_hide**(**struct** *hd44780_driver_t* *self_p)
Do not show the text on the display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_scroll_left**(**struct** *hd44780_driver_t* *self_p)
Scroll the text on the display one position to the left.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **hd44780_scroll_right**(**struct** *hd44780_driver_t *self_p*)

Scroll the text on the display one position to the right.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

**struct hd44780_driver_t**

### Public Members

**struct** pin_device_t ***rs_p**

**struct** pin_device_t ***enable_p**

**struct** pin_device_t ***data_4_p**

**struct** pin_device_t ***data_5_p**

**struct** pin_device_t ***data_6_p**

**struct** pin_device_t ***data_7_p**

unsigned int **number_of_rows**

unsigned int **number_of_columns**

unsigned int **row**

unsigned int **column**

**struct hd44780_driver_t::@0 hd44780_driver_t::cursor**

uint8_t **display_on_off_control**

## `led_7seg_ht16k33` — LED 7-Segment HT16K33

This is a driver for **'Adafruit 0.56" 4-Digit 7-Segment Display w/I2C Backpack'_** or compatible devices which uses the Holtek HT16K33 chip.

At this time the driver only supports using the *i2c_soft — Software I2C* driver to communicate with the HT16K33, not the *i2c — I2C* driver.

Source code: src/drivers/displays/led_7seg_ht16k33.h, src/drivers/displays/led_7seg_ht16k33.c

---

### Defines

**LED_7SEG_HT16K33_BRIGHTNESS_MIN**
Minimum brightness.

**LED_7SEG_HT16K33_BRIGHTNESS_MAX**
Maximum brightness.

**LED_7SEG_HT16K33_DEFAULT_I2C_ADDR**
Default I2C address for HT16K33.

### Functions

int **led_7seg_ht16k33_module_init**(void)
Initialize the driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **led_7seg_ht16k33_init**(**struct** *led_7seg_ht16k33_driver_t* **self_p*, **struct** *i2c_soft_driver_t* **i2c_p*, int *i2c_addr*)
Initialize driver object. The driver object will be used for a single display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialize.
- `i2c_p`: The I2C driver pointer.
- `i2c_addr`: The address of the HT16K33 controller. Probably LED_7SEG_HT16K33_DEFAULT_I2C_ADDR.

int **led_7seg_ht16k33_start**(**struct** *led_7seg_ht16k33_driver_t* **self_p*)
Start driver.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **led_7seg_ht16k33_display**(**struct** *led_7seg_ht16k33_driver_t* **self_p*)
Send content of display buffer to the display.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **led_7seg_ht16k33_clear**(**struct** *led_7seg_ht16k33_driver_t* **self_p*)
Clear content of display buffer.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.

int **led_7seg_ht16k33_brightness**(**struct** *led_7seg_ht16k33_driver_t* **self_p*, int *brightness*)
Set display brightness.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `brightness`: Brightness from LED_7SEG_HT16K33_BRIGHTNESS_MIN to LED_7SEG_HT16K33_BRIGHTNESS_MAX.

int **led_7seg_ht16k33_set_num**(**struct** *led_7seg_ht16k33_driver_t* **self_p*, int *num*, int *base*)

Set a number in the display buffer.

Number cannot be more than 4 digits AKA base^4 - 1.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `num`: Number to set.
- `base`: Base of *num*.

int **led_7seg_ht16k33_show_colon**(**struct** *led_7seg_ht16k33_driver_t* **self_p*, int *show_colon*)

Set show/hide of colon in the display buffer.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `show_colon`: If true light colon, otherwise turn off.

int **led_7seg_ht16k33_show_dot**(**struct** *led_7seg_ht16k33_driver_t* **self_p*, int *position*, int *show_colon*)

Set show/hide of dot in the display buffer.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `position`: The position to light colon or not. Range: 0 to 3.
- `show_dot`: If true light dot, otherwise turn off.

**struct led_7seg_ht16k33_driver_t**

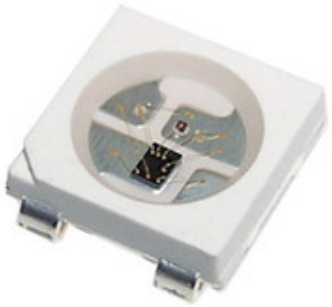
### Public Members

**struct** *i2c_soft_driver_t* ***i2c_p**

int **i2c_addr**

uint8_t **buf**[5]

### ws2812 — NeoPixels



WS2812 is a NeoPixel. Source code: src/drivers/displays/ws2812.h, src/drivers/displays/ws2812.c

---

## Defines

**WS2812_PIN_DEVICES_MAX**
Maximum number of pin devices a driver object can handle.

## Functions

int **ws2812_module_init**(void)
Initialize the WS2812 driver module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **ws2812_init**(**struct** *ws2812_driver_t* **self_p*, **struct** pin_device_t ***pin_devices_pp*, int *number_of_pin_devices*)
Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `pin_devices_pp`: An array of pin device(s) to use. The maximum length of the array is defined as `WS2812_PIN_DEVICES_MAX`.
- `number_of_pin_devices`: Number of pin devices in the pin devices array.

int **ws2812_write**(**struct** *ws2812_driver_t* **self_p*, **const** uint8_t **colors_p*, int *number_of_pixles*)
Write given RGB colors to the NeoPixels.

CAUTION: Interrupts are disabled during the write to meet the strict timing requirements on the pulse train. It takes ~30 us to write to one pixel, that is, interrupts are disabled for ~30 * number_of_pixles us. Long pixel chains may cause the rest of the system to misbehave.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `colors_p`: An array of GRB colors to write to the NeoPixels. All pin devices green component first, then all red, and last all blue, repeated for all NeoPixels. For example, when a single pin device is configured the array is G0, R0, B0, G1, R1, B1, ...
- `number_of_pixles`: Number of GRB colors per pin device in `colors_p`.

**`struct ws2812_driver_t`**

#### Public Members

**`struct`** pin_device_t ****`pins_pp`**

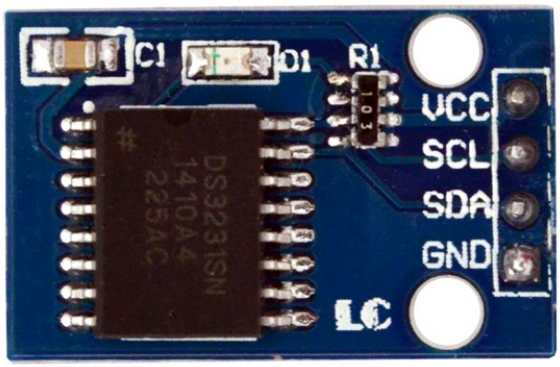
int **`number_of_pins`**

uint32_t **`mask`**

### various

Various drivers that does not fit into any other category.

#### `ds3231` — RTC clock




DS3231 is a real time clock. Source code: src/drivers/various/ds3231.h, src/drivers/various/ds3231.c

Test code: tst/drivers/hardware/various/ds3231/main.c

### Functions

int **ds3231_init**(**struct** *ds3231_driver_t* *self_p*, **struct** i2c_driver_t *i2c_p*)
Initialize given driver object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `i2c_p`: I2C driver to use.

int **ds3231_set_date**(**struct** *ds3231_driver_t* *self_p*, **struct** *date_t* *date_p*)
Set date in the DS3231 device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `date_p`: Date to set in the device.

int **ds3231_get_date**(**struct** *ds3231_driver_t* *self_p*, **struct** *date_t* *date_p*)
Get date from the DS3231 device.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `date_p`: Date read from the device.

**struct ds3231_driver_t**

#### Public Members

**struct** i2c_driver_t ***i2c_p**

## gnss — Global Navigation Satellite System

GNSS is a set of global coverage satellite system, including GPS, GLONASS and Galileo.

This driver reads NMEA 0183 sentences from an input channel (often a UART driver), parses them and stores position, time and speed in the driver object.

This driver should be compatible with all GNSS devices sending and receiving NMEA sentences over a serial port.

Devices known to work with this driver:

-

---

Source code: src/drivers/various/gnss.h, src/drivers/various/gnss.c

Test code: tst/drivers/software/various/gnss/main.c

Example code: examples/gnss/main.c

### Functions

int **gnss_module_init**(void)
Initialize the GNSS module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **gnss_init**(**struct** *gnss_driver_t* **self_p*, void **chin_p*, void **chout_p*)
Initialize given driver object from given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to be initialized.
- `chin_p`: Incoming transport channel.
- `chout_p`: Outgoing transport channel.

int **gnss_read**(**struct** *gnss_driver_t* **self_p*)
Update the GNSS driver state by reading and parsing a NMEA sentence from the transport channel.

NOTE: NMEA sentences will be lost if this function is called too seldom (due to transport channel input overrun).

**Return** zero(0) if an NEMA sentence was read, otherwise negative error code.

**Parameters**

- `self_p`: Initialized driver object.

int **gnss_write**(**struct** *gnss_driver_t* **self_p*, char **str_p*)
Encode an NMEA sentence of given string by prepending a `$` and appending a CRC and line termination, and write the sentence to the transport channel.

For example, the string `GPGLL,4916.45,N,12311.12,W,225444,A,` is encoded to `$GPGLL,4916.45,N,12311.12,W,225444,A,*1D\r\n` and then written to the transport channel.

**Return** zero(0) if the NEMA sentence was written to the transport channel, otherwise negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `str_p`: String to write.

int **gnss_get_date**(**struct** *gnss_driver_t* **self_p*, **struct** *date_t* **date_p*)
Get most recently received date.

**Return** Date age in seconds or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `date_p`: Current date.

int **gnss_get_position**(**struct** *gnss_driver_t* *self_p, float *latitude_p, float *longitude_p)
Get most recently received position.

**Return** Position age in seconds or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `latitude_p`: Current latitude in degrees. Positive number for north, negative south.
- `longitude_p`: Current longitude in degrees. Positive number for east, negative for west.

int **gnss_get_speed**(**struct** *gnss_driver_t* *self_p, float *speed_p)
Get most recently received speed.

**Return** Speed age in seconds or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `speed_p`: Current speed in meters per second.

int **gnss_get_number_of_satellites**(**struct** *gnss_driver_t* *self_p, int *number_of_satellites_p)
Get most recently received number of tracked satellites.

**Return** Number of tracked satellites age in seconds or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `number_of_satellites_p`: Current number of tracked satellites.

int **gnss_get_altitude**(**struct** *gnss_driver_t* *self_p, float *altitude_p)
Get most recently received altitude.

**Return** Altitude age in seconds or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `altitude_p`: Current altitude in meters.

int **gnss_print**(**struct** *gnss_driver_t* *self_p, void *chan_p)
Print the driver state as a human readable string to given channel.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized driver object.
- `chan_p`: Channel to print to.

**struct gnss_driver_t**

#### Public Members

void *chin_p

void *chout_p

struct *time_t* rmc_timestamp

struct *time_t* gga_timestamp

struct *date_t* date

long latitude_degrees

long longitude_degrees

struct *time_t* *timestamp_p

struct gnss_driver_t::@42 gnss_driver_t::position

long speed

int number_of_satellites

long altitude

char buf[NMEA_SENTENCE_SIZE_MAX]

size_t size

struct gnss_driver_t::@43::@44 gnss_driver_t::input

struct *nmea_sentence_t* decoded

struct gnss_driver_t::@43 gnss_driver_t::nmea

struct *log_object_t* log

### 1.6.4 filesystems

File systems and file system like frameworks.

The filesystems package on Github.

#### fat16 — FAT16 filesystem

File Allocation Table (FAT) is a computer file system architecture and a family of industry-standard file systems utilizing it. The FAT file system is a legacy file system which is simple and robust. It offers good performance even in light-weight implementations, but cannot deliver the same performance, reliability and scalability as some modern file systems. It is, however, supported for compatibility reasons by nearly all currently developed operating systems for personal computers and many mobile devices and embedded systems, and thus is a well-suited format for data exchange between computers and devices of almost any type and age from 1981 up to the present.

#### Example

Here is the pseudo-code for mounting a file system, performing file operations and unmounting the file system.

All function arguments are omitted in this example.

```
/* Mount the file system. This is normally done once when the
   application starts. */
fat16_init();
fat16_mount();

/* Perform file operations. */
fat16_file_open();
fat16_file_read();
fat16_file_close();

fat16_file_open();
fat16_file_write();
fat16_file_close();

/* Unmount the file system when it is no long needed. Normally when
   the application stops. */
fat16_unmount();
```

---

Source code: src/filesystems/fat16.h, src/filesystems/fat16.c

Test code: tst/filesystems/fat16/main.c

Test coverage: src/filesystems/fat16.c

Example code: examples/fat16/main.c

---

### Defines

**FAT16_SEEK_SET**
The offset is relative to the start of the file.

**FAT16_SEEK_CUR**
The offset is relative to the current position indicator.

**FAT16_SEEK_END**
The offset is relative to the end of the file.

**FAT16_EOF**
End of file indicator.

**O_READ**
Open for reading.

**O_RDONLY**
Same as O_READ.

**O_WRITE**
Open for write.

**O_WRONLY**
Same as O_WRITE.

**O_RDWR**
Open for reading and writing.

**O_APPEND**
The file position indicator shall be set to the end of the file prior to each write.

**O_SYNC**
Synchronous writes.

**O_CREAT**
Create the file if non-existent.

**O_EXCL**
If O_CREAT and O_EXCL are set, file open shall fail if the file exists.

**O_TRUNC**
Truncate the file to zero length.

**DIR_ATTR_READ_ONLY**
File is read-only.

**DIR_ATTR_HIDDEN**
File should hidden in directory listings.

**DIR_ATTR_SYSTEM**
Entry is for a system file.

**DIR_ATTR_VOLUME_ID**
Directory entry contains the volume label.

**DIR_ATTR_DIRECTORY**
Entry is for a directory.

**DIR_ATTR_ARCHIVE**
Old DOS archive bit for backup support.

### Typedefs

**typedef** ssize_t (***fat16_read_t**)(void *arg_p, void *dst_p, uint32_t src_block)
Block read function callback.

**typedef** ssize_t (***fat16_write_t**)(void *arg_p, uint32_t dst_block, **const** void *src_p)
Block write function callback.

**typedef** uint16_t **fat_t**
A FAT entry.

### Functions

int **fat16_init**(**struct** *fat16_t *self_p*, *fat16_read_t read*, *fat16_write_t write*, void **arg_p*, unsigned int *partition*)
Initialize a FAT16 volume.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: FAT16 object to initialize.
- `read`: Callback function used to read blocks of data.
- `write`: Callback function used to write blocks of data.
- `arg_p`: Argument passed as the first arguemtn to read() and write().

- `partition`: Partition to be used. Legal values for a partition are 1-4 to use the corresponding partition on a device formatted with a MBR, Master Boot Record, or zero if the device is formatted as a super floppy with the FAT boot sector in block zero.

int **fat16_mount**(**struct** *fat16_t *self_p*)
Mount given FAT16 volume.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: FAT16 object.

int **fat16_unmount**(**struct** *fat16_t *self_p*)
Unmount given FAT16 volume.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: FAT16 object.

int **fat16_format**(**struct** *fat16_t *self_p*)
Create an empty FAT16 file system on the device.

**Parameters**

- `self_p`: FAT16 object.

int **fat16_print**(**struct** *fat16_t *self_p*, void *chan_p*)
Print volume information to given channel.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: FAT16 object.
- `chan_p`: Output channel.

int **fat16_file_open**(**struct** *fat16_t *self_p*, **struct** *fat16_file_t *file_p*, **const** char *path_p, int *oflag*)
Open a file by file path and mode flags.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: FAT16 object.
- `file_p`: File object to be initialized.
- `path_p`: A valid 8.3 DOS name for a file path.
- `oflag`: mode of file open (create, read, write, etc).

int **fat16_file_close**(**struct** *fat16_file_t *file_p*)
Close a file and force cached data and directory information to be written to the media.

**Return** zero(0) or negative error code.

**Parameters**

- `file_p`: File object.

ssize_t **fat16_file_read**(**struct** *fat16_file_t* *file_p, void *buf_p, size_t *size*)
Read data to given buffer with given size from the file.

**Return** Number of bytes read or EOF(-1).

**Parameters**

- `file_p`: File object.
- `buf_p`: Buffer to read into.
- `size`: number of bytes to read.

ssize_t **fat16_file_write**(**struct** *fat16_file_t* *file_p, **const** void *buf_p, size_t *size*)
Write data from buffer with given size to the file.

**Return** Number of bytes written or EOF(-1).

**Parameters**

- `file_p`: File object.
- `buf_p`: Buffer to write.
- `size`: number of bytes to write.

int **fat16_file_seek**(**struct** *fat16_file_t* *file_p, int *pos*, int *whence*)
Sets the file's read/write position relative to mode.

**Return** zero(0) or negative error code.

**Parameters**

- `file_p`: File object.
- `pos`: New position in bytes from given mode.
- `whence`: Absolute, relative or from end.

ssize_t **fat16_file_tell**(**struct** *fat16_file_t* *file_p)
Return current position in the file.

**Return** Current position or negative error code.

**Parameters**

- `file_p`: File object.

int **fat16_file_truncate**(**struct** *fat16_file_t* *file_p, size_t *size*)
Truncate given file to a size of precisly `size` bytes.

If the file previously was larger than this size, the extra data is lost. If the file previously was shorter, it is extended, and the extended part reads as null bytes ('\0').

**Return** zero(0) or negative error code.

**Parameters**

- `file_p`: File object.

- `size`: New size of the file in bytes.

ssize_t **fat16_file_size**(**struct** *fat16_file_t *file_p*)
Return number of bytes in the file.

**Return** File size in bytes or negative error code.

**Parameters**

- `file_p`: File object.

int **fat16_file_sync**(**struct** *fat16_file_t *file_p*)
Causes all modified data and directory fields to be written to the storage device.

**Return** zero(0) or negative error code.

**Parameters**

- `file_p`: File object.

int **fat16_dir_open**(**struct** *fat16_t *self_p*, **struct** *fat16_dir_t *dir_p*, **const** char *path_p*, int *oflag*)
Open a directory by directory path and mode flags.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: FAT16 object.
- `dir_p`: Directory object to be initialized.
- `path_p`: A valid 8.3 DOS name for a directory path.
- `oflag`: mode of the directory to open (create, read, etc).

int **fat16_dir_close**(**struct** *fat16_dir_t *dir_p*)
Close given directory.

**Return** zero(0) or negative error code.

**Parameters**

- `dir_p`: Directory object.

int **fat16_dir_read**(**struct** *fat16_dir_t *dir_p*, **struct** *fat16_dir_entry_t *entry_p*)
Read the next file or directory within the opened directory.

**Return** true(1) if an entry was read or false(0) if no entry could be read, otherwise negative error code.

**Parameters**

- `dir_p`: Directory object.
- `entry_p`: Read entry.

int **fat16_stat**(**struct** *fat16_t *self_p*, **const** char *path_p*, **struct** *fat16_stat_t *stat_p*)
Gets file status by path.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `path_p`: The path of the file to stat.
- `stat_p`: The stat struct to populate.

### Variables

**struct** *dir_t* **PACKED**

**union fat16_time_t**

*#include <fat16.h>* FAT Time Format. A FAT directory entry time stamp is a 16-bit field that has a granularity of 2 seconds. Here is the format (bit 0 is the LSB of the 16-bit word, bit 15 is the MSB of the 16-bit word).

Bits 0-4: 2-second count, valid value range 0-29 inclusive (0-58 seconds). Bits 5-10: Minutes, valid value range 0-59 inclusive. Bits 11-15: Hours, valid value range 0-23 inclusive.

The valid time range is from Midnight 00:00:00 to 23:59:58.

#### Public Members

uint16_t **as_uint16**

uint16_t **seconds**

uint16_t **minutes**

uint16_t **hours**

**struct fat16_time_t::@48 fat16_time_t::bits**

**union fat16_date_t**

*#include <fat16.h>* FAT date representation support Date Format. A FAT directory entry date stamp is a 16-bit field that is basically a date relative to the MS-DOS epoch of 01/01/1980. Here is the format (bit 0 is the LSB of the 16-bit word, bit 15 is the MSB of the 16-bit word):

Bits 0-4: Day of month, valid value range 1-31 inclusive. Bits 5-8: Month of year, 1 = January, valid value range 1-12 inclusive. Bits 9-15: Count of years from 1980, valid value range 0-127 inclusive (1980-2107).

#### Public Members

uint16_t **as_uint16**

uint16_t **day**

uint16_t **month**

uint16_t **year**

**struct fat16_date_t::@49 fat16_date_t::bits**

**struct part_t**

*#include <fat16.h>* MBR partition table entry. A partition table entry for a MBR formatted storage device. The MBR partition table has four entries.

### Public Members

uint8_t **boot**
: Boot Indicator. Indicates whether the volume is the active partition. Legal values include: 0x00. Do not use for booting. 0x80 Active partition.

uint8_t **begin_head**
: Head part of Cylinder-head-sector address of the first block in the partition. Legal values are 0-255. Only used in old PC BIOS.

unsigned **begin_sector**
: Sector part of Cylinder-head-sector address of the first block in the partition. Legal values are 1-63. Only used in old PC BIOS.

unsigned **begin_cylinder_high**
: High bits cylinder for first block in partition.

uint8_t **begin_cylinder_low**
: Combine beginCylinderLow with beginCylinderHigh. Legal values are 0-1023. Only used in old PC BIOS.

uint8_t **type**
: Partition type. See defines that begin with PART_TYPE_ for some Microsoft partition types.

uint8_t **end_head**
: head part of cylinder-head-sector address of the last sector in the partition. Legal values are 0-255. Only used in old PC BIOS.

unsigned **end_sector**
: Sector part of cylinder-head-sector address of the last sector in the partition. Legal values are 1-63. Only used in old PC BIOS.

unsigned **end_cylinder_high**
: High bits of end cylinder

uint8_t **end_cylinder_low**
: Combine endCylinderLow with endCylinderHigh. Legal values are 0-1023. Only used in old PC BIOS.

uint32_t **first_sector**
: Logical block address of the first block in the partition.

uint32_t **total_sectors**
: Length of the partition, in blocks.

**struct bpb_t**
: *#include <fat16.h>* BIOS parameter block; The BIOS parameter block describes the physical layout of a FAT volume.

### Public Members

uint16_t **bytes_per_sector**
: Count of bytes per sector. This value may take on only the following values: 512, 1024, 2048 or 4096

uint8_t **sectors_per_cluster**
: Number of sectors per allocation unit. This value must be a power of 2 that is greater than 0. The legal values are 1, 2, 4, 8, 16, 32, 64, and 128.

uint16_t **reserved_sector_count**
: Number of sectors before the first FAT. This value must not be zero.

uint8_t **fat_count**
The count of FAT data structures on the volume. This field should always contain the value 2 for any FAT volume of any type.

uint16_t **root_dir_entry_count**
For FAT12 and FAT16 volumes, this field contains the count of 32-byte directory entries in the root directory. For FAT32 volumes, this field must be set to 0. For FAT12 and FAT16 volumes, this value should always specify a count that when multiplied by 32 results in a multiple of bytesPerSector. FAT16 volumes should use the value 512.

uint16_t **total_sectors_small**
This field is the old 16-bit total count of sectors on the volume. This count includes the count of all sectors in all four regions of the volume. This field can be 0; if it is 0, then totalSectors32 must be non-zero. For FAT32 volumes, this field must be 0. For FAT12 and FAT16 volumes, this field contains the sector count, and totalSectors32 is 0 if the total sector count fits (is less than 0x10000).

uint8_t **media_type**
This dates back to the old MS-DOS 1.x media determination and is no longer usually used for anything. 0xf8 is the standard value for fixed (non-removable) media. For removable media, 0xf0 is frequently used. Legal values are 0xf0 or 0xf8-0xff.

uint16_t **sectors_per_fat**
Count of sectors occupied by one FAT on FAT12/FAT16 volumes. On FAT32 volumes this field must be 0, and sectorsPerFat32 contains the FAT size count.

uint16_t **sectors_per_track**
Sectors per track for interrupt 0x13. Not used otherwise.

uint16_t **head_count**
Number of heads for interrupt 0x13. Not used otherwise.

uint32_t **hiddden_sectors**
Count of hidden sectors preceding the partition that contains this FAT volume. This field is generally only relevant for media visible on interrupt 0x13.

uint32_t **total_sectors_large**
This field is the new 32-bit total count of sectors on the volume. This count includes the count of all sectors in all four regions of the volume. This field can be 0; if it is 0, then totalSectors16 must be non-zero.

**struct fbs_t**
*#include* <*fat16.h*> Boot sector for a FAT16 or FAT32 volume.

### Public Members

uint8_t **jmp_to_boot_code**[3]
X86 jmp to boot program

char **oem_name**[8]
Informational only - don't depend on it

**struct** *bpb_t* **bpb**
BIOS Parameter Block

uint8_t **drive_number**
For int0x13 use value 0x80 for hard drive

uint8_t **reserved1**
Used by Windows NT - should be zero for FAT

uint8_t **boot_signature**
0x29 if next three fields are valid

uint32_t **volume_serial_number**
Usually generated by combining date and time

char **volume_label**[11]
Should match volume label in root dir

char **file_system_type**[8]
Informational only - don't depend on it

uint8_t **boot_code**[448]
X86 boot code

uint16_t **boot_sector_sig**
Must be 0x55AA

**struct mbr_t**
*#include <fat16.h>* Master Boot Record. The first block of a storage device that is formatted with a MBR.

### Public Members

uint8_t **codeArea**[440]
Code Area for master boot program.

uint32_t **diskSignature**
Optional WindowsNT disk signature. May contain more boot code.

uint16_t **usuallyZero**
Usually zero but may be more boot code.

**struct** *part_t* **part**[4]
Partition tables.

uint16_t **mbr_sig**
First MBR signature byte. Must be 0x55

**struct dir_t**
*#include <fat16.h>* FAT short directory entry. Short means short 8.3 name, not the entry size.

### Public Members

uint8_t **name**[11]
Short 8.3 name. The first eight bytes contain the file name with blank fill. The last three bytes contain the file extension with blank fill.

uint8_t **attributes**
Entry attributes. The upper two bits of the attribute byte are reserved and should always be set to 0 when a file is created and never modified or looked at after that. See defines that begin with DIR_ATT_.

uint8_t **reserved1**
Reserved for use by Windows NT. Set value to 0 when a file is created and never modify or look at it after that.

uint8_t **creation_time_tenths**
The granularity of the seconds part of creationTime is 2 seconds so this field is a count of tenths of a second and its valid value range is 0-199 inclusive. (WHG note - seems to be hundredths)

uint16_t **creation_time**
Time file was created.

uint16_t **creation_date**
Date file was created.

uint16_t **last_access_date**
Last access date. Note that there is no last access time, only a date. This is the date of last read or write. In the case of a write, this should be set to the same date as lastWriteDate.

uint16_t **first_cluster_high**
High word of this entry's first cluster number (always 0 for a FAT12 or FAT16 volume).

uint16_t **last_write_time**
Time of last write. File creation is considered a write.

uint16_t **last_write_date**
Date of last write. File creation is considered a write.

uint16_t **first_cluster_low**
Low word of this entry's first cluster number.

uint32_t **file_size**
32-bit unsigned holding this file's size in bytes.

**union fat16_cache16_t**

### Public Members

uint8_t **data**[512]

*fat_t* **fat**[256]

**struct** *dir_t* **dir**[16]

**struct** *mbr_t* **mbr**

**struct** *fbs_t* **fbs**

**struct fat16_cache_t**

### Public Members

uint32_t **block_number**

uint8_t **dirty**

uint32_t **mirror_block**

**union** *fat16_cache16_t* **buffer**

**struct fat16_t**

### Public Members

*fat16_read_t* **read**

*fat16_write_t* **write**

void ***arg_p**

unsigned int **partition**

uint8_t **fat_count**

uint8_t **blocks_per_cluster**

uint16_t **root_dir_entry_count**

*fat_t* **blocks_per_fat**

*fat_t* **cluster_count**

uint32_t **volume_start_block**

uint32_t **fat_start_block**

uint32_t **root_dir_start_block**

uint32_t **data_start_block**

**struct** *fat16_cache_t* **cache**

**struct fat16_file_t**

#### Public Members

**struct** *fat16_t* ***fat16_p**

uint8_t **flags**

int16_t **dir_entry_block**

int16_t **dir_entry_index**

*fat_t* **first_cluster**

size_t **file_size**

*fat_t* **cur_cluster**

size_t **cur_position**

**struct fat16_dir_t**

#### Public Members

int16_t **root_index**

**struct** *fat16_file_t* **file**

**struct fat16_dir_entry_t**

#### Public Members

char **name**[256]

int **is_dir**

size_t **size**

**struct** *date_t* **latest_mod_date**

**struct fat16_stat_t**

#### Public Members

size_t **size**

int **is_dir**

### fs — Debug and virtual file system

The debug file system is not really a file system, but rather a file system like tree of commands, counters, parameters, and "real" file systems.

- A command is a file path mapped to a function callback. The callback is invoked when its path is passed to the `fs_call()` function. Commands are registered into the debug file system by a call to `fs_command_register()`.
- A counter is a file path mapped to a 64 bit value. The value can be incremented and read by the application. Counters are registered into the debug file system by a call to `fs_counter_register()`.
- A parameter is file path mapped to a value stored in ram that can be easily read and modified by the user from a shell. Parameters are registered into the debug file system by a call to `fs_parameter_register()`.
- A "real" file system is a file path, or mount point, mapped to a file system instance. The virtual file system has a file access interface. The purpose of this interface is to have a common file access interface, independent of the underlying file systems interface. File systems are registered into the debug file system by a call to `fs_filesystem_register()`.

### Debug file system commands

The debug file system module itself registers seven commands, all located in the directory `filesystems/fs/`.

| Command | Description |
|---|---|
| `filesystems/list` | Print a list of all registered file systems. |
| `list [<folder>]` | Print a list of all files and folders in given folder. |
| `read <file>` | Read from given file. |
| `write <file> <data>` | Create and write to a file. Overwrites existing files. |
| `append <file> <data>` | Append data to an existing file. |
| `counters/list` | Print a list of all registered counters. |
| `counters/reset` | Rest all counters to zero. |
| `parameters/list` | Print a list of all registered parameters. |

Example output from the shell:

```
$ filesystems/fs/filesystems/list
MOUNT-POINT                    MEDIUM   TYPE     AVAILABLE  SIZE  USAGE
/tmp                           ram      fat16          54K   64K    14%
/home/erik                     sd       fat16        1.9G    2G     5%
/etc                           flash    spiffs       124K  128K     3%
OK
$ filesystems/fs/write tmp/foo.txt "Hello "
OK
$ filesystems/fs/append tmp/foo.txt world!
OK
$ filesystems/fs/read tmp/foo.txt
Hello world!
```

(continues on next page)

(continued from previous page)

```
OK
$ filesystems/fs/list tmp
xxxx-xx-xx xx-xx       12 foo.txt
OK
$ filesystems/fs/counters/list
NAME                                                    VALUE
/your/counter                                           0000000000000034
/my/counter                                             0000000000000002
OK
$ filesystems/fs/counters/reset
OK
$ filesystems/fs/counters/list
NAME                                                    VALUE
/your/counter                                           0000000000000000
/my/counter                                             0000000000000000
OK
$ filesystems/fs/parameters/list
NAME                                                    VALUE
/foo/bar                                                -2
OK
```

---

Source code: src/filesystems/fs.h, src/filesystems/fs.c

Test code: tst/filesystems/fs/main.c

Test coverage: src/filesystems/fs.c

---

### Defines

**FS_SEEK_SET**
The offset is relative to the start of the file.

**FS_SEEK_CUR**
The offset is relative to the current position indicator.

**FS_SEEK_END**
The offset is relative to the end of the file.

**FS_READ**
Open for reading.

**FS_WRITE**
Open for write.

**FS_RDWR**
Open for reading and writing.

**FS_APPEND**
The file position indicator shall be set to the end of the file prior to each write.

**FS_SYNC**
Synchronous writes.

**FS_CREAT**
Create the file if non-existent.

**FS_EXCL**
If FS_CREAT and FS_EXCL are set, file open shall fail if the file exists.

**FS_TRUNC**
Truncate the file to zero length.

**FS_TYPE_FILE**

**FS_TYPE_DIR**

**FS_TYPE_HARD_LINK**

**FS_TYPE_SOFT_LINK**

### Typedefs

**typedef** int (***fs_callback_t**) (int argc, **const** char *argv[], void *out_p, void *in_p, void *arg_p, void *call_arg_p)
Command callback prototype.

**Return** zero(0) or negative error code.

**Parameters**

- `argc`: Number of arguements in argv.
- `argv`: An array of agruments.
- `out_p`: Output channel.
- `in_p`: Input channel.
- `arg_p`: Argument passed to the init function of given command.
- `call_arg_p`: Argument passed to the `fs_call` function.

**typedef** int (***fs_parameter_set_callback_t**) (void *value_p, **const** char *src_p)
Parameter setter callback prototype.

**Return** zero(0) or negative error code.

**Parameters**

- `value_p`: Buffer the new value should be written to.
- `src_p`: Value to set as a string.

**typedef** int (***fs_parameter_print_callback_t**) (void *chout_p, void *value_p)
Parameter printer callback prototype.

**Return** zero(0) or negative error code.

**Parameters**

- `chout_p`: Channel to write the formatted value to.
- `value_p`: Value to format and print to the output channel.

### Enums

**enum fs_type_t**

*Values:*

**fs_type_fat16_t** = 0

**fs_type_spiffs_t**

**fs_type_generic_t**

### Functions

int **fs_module_init**(void)

Initialize the file system module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **fs_call**(char **command_p*, void **chin_p*, void **chout_p*, void **arg_p*)

Call given file system command with given input and output channels. Quote an argument if it contains spaces, otherwise it is parsed as multiple arguments. Any quotation mark in an argument string must be escaped with a backslash (\), otherwise it is interpreted as a string quotation mask.

**Return** zero(0) or negative error code.

**Parameters**

- `command_p`: Command string to call. The command string will be modified by this function, so don't use it after this function returns.
- `chin_p`: Input channel.
- `chout_p`: Output channel.
- `arg_p`: User argument passed to the command callback function as `call_arg_p`.

int **fs_open**(**struct** *fs_file_t* **self_p*, **const** char **path_p*, int *flags*)

Open a file by file path and mode flags. File operations are permitted after the file has been opened.

The path can be either absolute or relative. It's an absolute path if it starts with a forward slash `/`, and relative otherwise. Relative paths are are relative to the current working directory, given by the thread environment variable `CWD`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: File object to be initialized.
- `path_p`: Path of the file to open. The path can be absolute or relative.
- `flags`: Mode of file open. A combination of `FS_READ`, `FS_RDONLY`, `FS_WRITE`, `FS_WRONLY`, `FS_RDWR`, `FS_APPEND`, `FS_SYNC`, `FS_CREAT`, `FS_EXCL` and `FS_TRUNC`.

int **fs_close**(**struct** *fs_file_t* **self_p*)

Close given file. No file operations are permitted on a closed file.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized file object.

ssize_t **fs_read**(**struct** *fs_file_t* *self_p, void *dst_p, size_t *size*)
Read from given file into given buffer.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Initialized file object.
- `dst_p`: Buffer to read data into.
- `size`: Number of bytes to read.

ssize_t **fs_read_line**(**struct** *fs_file_t* *self_p, void *dst_p, size_t *size*)
Read one line from given file into given buffer. The function reads one character at a time from given file until the destination buffer is full, a newline `\n` is found or end of file is reached.

**Return** If a line was found the number of bytes read not including the null-termination is returned. If the destination buffer becomes full before a newline character, the destination buffer size is returned. Otherwise a negative error code is returned.

**Parameters**

- `self_p`: Initialized file object.
- `dst_p`: Buffer to read data into. Should fit the whole line and null-termination.
- `size`: Size of the destination buffer.

ssize_t **fs_write**(**struct** *fs_file_t* *self_p, **const** void *src_p, size_t *size*)
Write from given buffer into given file.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Initialized file object.
- `src_p`: Buffer to write.
- `size`: Number of bytes to write.

int **fs_seek**(**struct** *fs_file_t* *self_p, int *offset*, int *whence*)
Sets the file's read/write position relative to whence.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized file object.
- `offset`: New position in bytes from given whence.
- `whence`: Absolute (`FS_SEEK_SET`), relative (`FS_SEEK_CUR`) or from end (`FS_SEEK_END`).

ssize_t **fs_tell**(**struct** *fs_file_t* *self_p)
Return current position in the file.

**Return** Current position or negative error code.

**Parameters**

- `self_p`: Initialized file object.

int **fs_dir_open**(**struct** *fs_dir_t* *dir_p, **const** char *path_p, int oflag)
Open a directory by directory path and mode flags.

**Return** zero(0) or negative error code.

**Parameters**

- `dir_p`: Directory object to be initialized.
- `path_p`: A valid path name for a directory path.
- `oflag`: mode of the directory to open (create, read, etc).

int **fs_dir_close**(**struct** *fs_dir_t* *dir_p)
Close given directory.

**Return** zero(0) or negative error code.

**Parameters**

- `dir_p`: Directory object.

int **fs_dir_read**(**struct** *fs_dir_t* *dir_p, **struct** *fs_dir_entry_t* *entry_p)
Read the next file or directory within the opened directory.

**Return** true(1) if an entry was read or false(0) if no entry could be read, otherwise negative error code.

**Parameters**

- `dir_p`: Directory object.
- `entry_p`: Read entry.

int **fs_remove**(**const** char *path_p)
Remove file by given path.

**Return** zero(0) or negative error code.

**Parameters**

- `path_p`: The path of the file to remove.

int **fs_stat**(**const** char *path_p, **struct** *fs_stat_t* *stat_p)
Gets file status by path.

**Return** zero(0) or negative error code.

**Parameters**

- `path_p`: The path of the file to stat.
- `stat_p`: The stat struct to populate.

int **fs_mkdir**(**const** char *path_p)
Craete a directory with given path.

**Return** zero(0) or negative error code.

**Parameters**

- `path_p`: The path of the directoy to create.

int **fs_format**(**const** char *_path_p_)
Format file system at given path.

**Return** zero(0) or negative error code.

**Parameters**

- `path_p`: The path to the root of the file system to format. All data in the file system will be deleted.

int **fs_ls**(**const** char *_path_p_, **const** char *_filter_p_, void *_chout_p_)
List files and folders in given path. Optionally with given filter. The list is written to the output channel.

**Return** zero(0) or negative error code.

**Parameters**

- `path_p`: Directory to list.
- `filter_p`: Filter out files and folders.
- `chout_p`: Output chan.

int **fs_list**(**const** char *_path_p_, **const** char *_filter_p_, void *_chout_p_)
List files (callbacks) and directories in given path. Optionally with given filter. The list is written to the output channel.

**Return** zero(0) or negative error code.

**Parameters**

- `path_p`: Directory to list.
- `filter_p`: Filter out files and folders.
- `chout_p`: Output chan.

int **fs_auto_complete**(char *_path_p_)
Auto-complete given path.

**Return** >=1 if completion happened. Number of autocompleted characters added to the path. 0 if no completion happend, or negative error code.

**Parameters**

- `path_p`: Absolute or relative path to auto-complete.

void **fs_split**(char *_buf_p_, char **_path_pp_, char **_cmd_pp_)
Split buffer into path and command inplace.

**Return** zero(0) or negative error code.

**Parameters**

- `buf_p`: Buffer to split.
- `path_pp`: Path or NULL if no path was found.

- `cmd_pp`: Command or empty string.

void **fs_merge**(char *_path_p_, char *_cmd_p_)
Merge path and command previously split using `fs_split()`.

**Return** zero(0) or negative error code.

**Parameters**

- `path_p`: Path from spilt.
- `cmd_p`: Command from split.

int **fs_filesystem_init_generic**(**struct** _fs_filesystem_t_ *_self_p_, **const** char *_name_p_, **struct** _fs_filesystem_operations_t_ *_ops_p_)
Initialize given generic file system.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: File system to initialize.
- `name_p`: Path to register.
- `ops_p`: File system function callbacks.

int **fs_filesystem_register**(**struct** _fs_filesystem_t_ *_self_p_)
Register given file system. Use the functions `fs_open()`, `fs_read()`, `fs_write()`, `fs_close()`, `fs_seek()`, fs_tell() and `fs_read_line()` to access files in a registerd file system.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: File system to register.

int **fs_filesystem_deregister**(**struct** _fs_filesystem_t_ *_self_p_)
Deregister given file system.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: File system to deregister.

int **fs_command_init**(**struct** _fs_command_t_ *_self_p_, far_string_t _path_p_, _fs_callback_t_ _callback_, void *_arg_p_)
Initialize given command.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Command to initialize.
- `path_p`: Path to register.
- `callback`: Command callback function.
- `arg_p`: Callback argument.

int **fs_command_register**(**struct** *fs_command_t *command_p*)
Register given command. Registered commands are called by the function `fs_call()`.

**Return** zero(0) or negative error code.

**Parameters**

- `command_p`: Command to register.

int **fs_command_deregister**(**struct** *fs_command_t *command_p*)
Deregister given command.

**Return** zero(0) or negative error code.

**Parameters**

- `command_p`: Command to deregister.

int **fs_counter_init**(**struct** *fs_counter_t *self_p*, far_string_t *path_p*, uint64_t *value*)
Initialize given counter.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Counter to initialize.
- `path_p`: Path to register.
- `value`: Initial value of the counter.

int **fs_counter_increment**(**struct** *fs_counter_t *self_p*, uint64_t *value*)
Increment given counter.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Command to initialize.
- `value`: Increment value.

int **fs_counter_register**(**struct** *fs_counter_t *counter_p*)
Register given counter.

**Return** zero(0) or negative error code.

**Parameters**

- `counter_p`: Counter to register.

int **fs_counter_deregister**(**struct** *fs_counter_t *counter_p*)
Deregister given counter.

**Return** zero(0) or negative error code.

**Parameters**

- `counter_p`: Counter to deregister.

int **fs_parameter_init**(**struct** *fs_parameter_t* *self_p, far_string_t path_p, *fs_parameter_set_callback_t* set_cb, *fs_parameter_print_callback_t* print_cb, void *value_p)

Initialize given parameter.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Parameter to initialize.
- `path_p`: Path to register.
- `set_cb`: Callback function set set the parameter value.
- `print_cb`: Callback function set print the parameter value.
- `value_p`: Value storage area.

int **fs_parameter_register**(**struct** *fs_parameter_t* *parameter_p)

Register given parameter.

**Return** zero(0) or negative error code.

**Parameters**

- `parameter_p`: Parameter to register.

int **fs_parameter_deregister**(**struct** *fs_parameter_t* *parameter_p)

Deregister given parameter.

**Return** zero(0) or negative error code.

**Parameters**

- `parameter_p`: Parameter to deregister.

int **fs_parameter_int_set**(void *value_p, **const** char *src_p)

Integer parameter setter function callback

**Return** zero(0) or negative error code.

**Parameters**

- `value_p`: Buffer the new value should be written to.
- `src_p`: Value to set as a string.

int **fs_parameter_int_print**(void *chout_p, void *value_p)

Integer parameter printer function callback

**Return** zero(0) or negative error code.

**Parameters**

- `chout_p`: Channel to write the formatted value to.
- `value_p`: Value to format and print to the output channel.

**struct fs_filesystem_t**

#### Public Members

**const** char ***name_p**

*fs_type_t* **type**

**struct** *fs_filesystem_operations_t* ***ops_p**

**struct fs_filesystem_t::@50::@52 fs_filesystem_t::generic**

**union fs_filesystem_t::@50 fs_filesystem_t::fs**

**union fs_filesystem_t::@51 fs_filesystem_t::config**

**struct** *fs_filesystem_t* ***next_p**

**struct fs_file_t**

#### Public Members

**struct** *fs_filesystem_t* ***filesystem_p**

**union fs_file_t::@53 fs_file_t::u**

**struct fs_stat_t**

*#include* <*fs.h*> Path stats.

#### Public Members

uint32_t **size**

uint8_t **type**

**struct fs_command_t**

#### Public Members

far_string_t **path_p**

*fs_callback_t* **callback**

void ***arg_p**

**struct** *fs_command_t* ***next_p**

**struct fs_counter_t**

#### Public Members

**struct** *fs_command_t* **command**

**long long unsigned int fs_counter_t::value**

**struct** *fs_counter_t* ***next_p**

**struct fs_parameter_t**

#### Public Members

**struct** *fs_command_t* **command**

*fs_parameter_set_callback_t* **set_cb**

*fs_parameter_print_callback_t* **print_cb**

void ***value_p**

**struct** *fs_parameter_t* ***next_p**

**struct fs_dir_t**

#### Public Members

**struct** *fs_filesystem_t* ***filesystem_p**

**union fs_dir_t::@54 fs_dir_t::u**

**struct fs_dir_entry_t**

#### Public Members

char **name**[256]

int **type**

size_t **size**

**struct** *date_t* **latest_mod_date**

**struct fs_filesystem_operations_t**

#### Public Members

int (***file_open**)(**struct** *fs_filesystem_t* *filesystem_p, **struct** *fs_file_t* *self_p, **const** char *path_p, int flags)

int (***file_close**)(**struct** *fs_file_t* *self_p)

ssize_t (***file_read**)(**struct** *fs_file_t* *self_p, void *dst_p, size_t size)

ssize_t (***file_write**)(**struct** *fs_file_t* *self_p, **const** void *src_p, size_t size)

int (***file_seek**)(**struct** *fs_file_t* *self_p, int offset, int whence)

ssize_t (***file_tell**)(**struct** *fs_file_t* *self_p)

### spiffs — SPI Flash File System

The source code of this module is based on https://github.com/pellepl/spiffs.

### About

Spiffs is a file system intended for SPI NOR flash devices on embedded targets.

Spiffs is designed with following characteristics in mind:

- Small (embedded) targets, sparse RAM without heap.
- Only big areas of data (blocks) can be erased.
- An erase will reset all bits in block to ones.
- Writing pulls one to zeroes.
- Zeroes can only be pulled to ones by erase.
- Wear leveling.

---

Source code: src/filesystems/spiffs.h, src/filesystems/spiffs.c

Test code: tst/filesystems/spiffs/main.c

---

### Defines

**SPIFFS_OK**

**SPIFFS_ERR_NOT_MOUNTED**

**SPIFFS_ERR_FULL**

**SPIFFS_ERR_NOT_FOUND**

**SPIFFS_ERR_END_OF_OBJECT**

**SPIFFS_ERR_DELETED**

**SPIFFS_ERR_NOT_FINALIZED**

**SPIFFS_ERR_NOT_INDEX**

**SPIFFS_ERR_OUT_OF_FILE_DESCS**

**SPIFFS_ERR_FILE_CLOSED**

**SPIFFS_ERR_FILE_DELETED**

**SPIFFS_ERR_BAD_DESCRIPTOR**

**SPIFFS_ERR_IS_INDEX**

**SPIFFS_ERR_IS_FREE**

**SPIFFS_ERR_INDEX_SPAN_MISMATCH**

**SPIFFS_ERR_DATA_SPAN_MISMATCH**

**SPIFFS_ERR_INDEX_REF_FREE**

**SPIFFS_ERR_INDEX_REF_LU**

**SPIFFS_ERR_INDEX_REF_INVALID**

**SPIFFS_ERR_INDEX_FREE**

**SPIFFS_ERR_INDEX_LU**

**SPIFFS_ERR_INDEX_INVALID**

**SPIFFS_ERR_NOT_WRITABLE**

**SPIFFS_ERR_NOT_READABLE**

**SPIFFS_ERR_CONFLICTING_NAME**

**SPIFFS_ERR_NOT_CONFIGURED**

**SPIFFS_ERR_NOT_A_FS**

**SPIFFS_ERR_MOUNTED**

**SPIFFS_ERR_ERASE_FAIL**

**SPIFFS_ERR_MAGIC_NOT_POSSIBLE**

**SPIFFS_ERR_NO_DELETED_BLOCKS**

**SPIFFS_ERR_FILE_EXISTS**

**SPIFFS_ERR_NOT_A_FILE**

**SPIFFS_ERR_RO_NOT_IMPL**

**SPIFFS_ERR_RO_ABORTED_OPERATION**

**SPIFFS_ERR_PROBE_TOO_FEW_BLOCKS**

**SPIFFS_ERR_PROBE_NOT_A_FS**

**SPIFFS_ERR_NAME_TOO_LONG**

**SPIFFS_ERR_INTERNAL**

**SPIFFS_ERR_TEST**

**SPIFFS_DBG**(...)

**SPIFFS_GC_DBG**(...)

**SPIFFS_CACHE_DBG**(...)

**SPIFFS_CHECK_DBG**(...)

**SPIFFS_APPEND**
Any write to the filehandle is appended to end of the file.

**SPIFFS_O_APPEND**

**SPIFFS_TRUNC**
If the opened file exists, it will be truncated to zero length before opened.

**SPIFFS_O_TRUNC**

**SPIFFS_CREAT**
If the opened file does not exist, it will be created before opened.

**SPIFFS_O_CREAT**

**SPIFFS_RDONLY**
The opened file may only be read.

**SPIFFS_O_RDONLY**

**SPIFFS_WRONLY**
The opened file may only be written.

**SPIFFS_O_WRONLY**

**SPIFFS_RDWR**
The opened file may be both read and written.

**SPIFFS_O_RDWR**

**SPIFFS_DIRECT**
Any writes to the filehandle will never be cached but flushed directly.

**SPIFFS_O_DIRECT**

**SPIFFS_EXCL**
If SPIFFS_O_CREAT and SPIFFS_O_EXCL are set, SPIFFS_open() shall fail if the file exists.

**SPIFFS_O_EXCL**

**SPIFFS_SEEK_SET**

**SPIFFS_SEEK_CUR**

**SPIFFS_SEEK_END**

**SPIFFS_TYPE_FILE**

**SPIFFS_TYPE_DIR**

**SPIFFS_TYPE_HARD_LINK**

**SPIFFS_TYPE_SOFT_LINK**

**SPIFFS_LOCK**(fs)

**SPIFFS_UNLOCK**(fs)

### Typedefs

**typedef** int16_t **spiffs_file_t**
Spiffs file descriptor index type. must be signed.

**typedef** uint16_t **spiffs_flags_t**
Spiffs file descriptor flags.

**typedef** uint16_t **spiffs_mode_t**
Spiffs file mode.

**typedef** uint8_t **spiffs_obj_type_t**
Object type.

**typedef** int32_t (***spiffs_read_cb_t**)(uint32_t addr, uint32_t size, uint8_t *dst_p)
Spi read call function type.

**typedef** int32_t (***spiffs_write_cb_t**)(uint32_t addr, uint32_t size, uint8_t *src_p)
Spi write call function type.

**typedef** int32_t (***spiffs_erase_cb_t**)(uint32_t addr, uint32_t size)
Spi erase call function type.

**typedef** void (***spiffs_check_callback_t**)(**enum** *spiffs_check_type_t* type, **enum** *spiffs_check_report_t* report, uint32_t arg1, uint32_t arg2)
File system check callback function.

**typedef** void (***spiffs_file_callback_t**)(**struct** *spiffs_t* *fs_p, **enum** *spiffs_fileop_type_t* op, spiffs_obj_id_t obj_id, spiffs_page_ix_t pix)

File system listener callback function.

**typedef** spiffs_block_ix_t **spiffs_block_ix**

**typedef** spiffs_page_ix_t **spiffs_page_ix**

**typedef** spiffs_obj_id_t **spiffs_obj_id**

**typedef** spiffs_span_ix_t **spiffs_span_ix**

**typedef struct** *spiffs_t* **spiffs**

**typedef** *spiffs_file_t* **spiffs_file**

**typedef** *spiffs_flags_t* **spiffs_flags**

**typedef** *spiffs_obj_type_t* **spiffs_obj_type**

**typedef** *spiffs_mode_t* **spiffs_mode**

**typedef enum** *spiffs_fileop_type_t* **spiffs_fileop_type**

**typedef struct** *spiffs_config_t* **spiffs_config**

**typedef** *spiffs_check_callback_t* **spiffs_check_callback**

**typedef struct** *spiffs_dirent_t* **spiffs_dirent**

**typedef struct** *spiffs_dir_t* **spiffs_DIR**

**typedef** *spiffs_file_callback_t* **spiffs_file_callback**

### Enums

**enum spiffs_check_type_t**

File system check callback report operation.

*Values:*

**SPIFFS_CHECK_LOOKUP** = 0

**SPIFFS_CHECK_INDEX**

**SPIFFS_CHECK_PAGE**

**enum spiffs_check_report_t**

File system check callback report type.

*Values:*

**SPIFFS_CHECK_PROGRESS** = 0

**SPIFFS_CHECK_ERROR**

**SPIFFS_CHECK_FIX_INDEX**

**SPIFFS_CHECK_FIX_LOOKUP**

**SPIFFS_CHECK_DELETE_ORPHANED_INDEX**

**SPIFFS_CHECK_DELETE_PAGE**

**SPIFFS_CHECK_DELETE_BAD_FILE**

**enum spiffs_fileop_type_t**
File system listener callback operation.

*Values:*

**SPIFFS_CB_CREATED** = 0
The file has been created.

**SPIFFS_CB_UPDATED**
The file has been updated or moved to another page.

**SPIFFS_CB_DELETED**
The file has been deleted.

### Functions

int32_t **spiffs_mount**(**struct** *spiffs_t* *self_p, **struct** *spiffs_config_t* *config_p, uint8_t *work_p, uint8_t *fd_space_p, uint32_t *fd_space_size*, void *cache_p, uint32_t *cache_size*, *spiffs_check_callback_t* *check_cb*)
Initializes the file system dynamic parameters and mounts the filesystem. If SPIFFS_USE_MAGIC is enabled the mounting may fail with SPIFFS_ERR_NOT_A_FS if the flash does not contain a recognizable file system. In this case, SPIFFS_format must be called prior to remounting.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `config_p`: The physical and logical configuration of the file system.
- `work_p`: A memory work buffer comprising 2*config->log_page_size bytes used throughout all file system operations
- `fd_space_p`: Memory for file descriptors.
- `fd_space_size`: Memory size of file descriptors.
- `cache_p`: Memory for cache, may be NULL.
- `cache_size`: Memory size of cache.
- `check_cb`: Callback function for reporting during consistency checks.

void **spiffs_unmount**(**struct** *spiffs_t* *self_p*)
Unmounts the file system. All file handles will be flushed of any cached writes and closed.

**Return** void.

**Parameters**

- `self_p`: The file system struct.

int32_t **spiffs_creat**(**struct** *spiffs_t* *self_p*, **const** char *path_p*, *spiffs_mode_t* *mode*)
Creates a new file.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.

- `path_p`: The path of the new file.
- `mode`: Ignored, for posix compliance.

*spiffs_file_t* **spiffs_open**(**struct** *spiffs_t* *self_p, **const** char *path_p, *spiffs_flags_t* flags, *spiffs_mode_t* mode)

Opens/creates a file.

**Parameters**

- `self_p`: The file system struct.
- `path_p`: The path of the new file.
- `flags`: The flags for the open command, can be combinations of SPIFFS_O_APPEND, SPIFFS_O_TRUNC, SPIFFS_O_CREAT, SPIFFS_O_RDONLY, SPIFFS_O_WRONLY, SPIFFS_O_RDWR, SPIFFS_O_DIRECT, SPIFFS_O_EXCL.
- `mode`: Ignored, for posix compliance.

*spiffs_file_t* **spiffs_open_by_dirent**(**struct** *spiffs_t* *self_p, **struct** *spiffs_dirent_t* *ent_p, *spiffs_flags_t* flags, *spiffs_mode_t* mode)

Opens a file by given dir entry.

Optimization purposes, when traversing a file system with SPIFFS_readdir a normal SPIFFS_open would need to traverse the filesystem again to find the file, whilst SPIFFS_open_by_dirent already knows where the file resides.

**Parameters**

- `self_p`: The file system struct.
- `e_p`: The dir entry to the file.
- `flags`: The flags for the open command, can be combinations of SPIFFS_APPEND, SPIFFS_TRUNC, SPIFFS_CREAT, SPIFFS_RD_ONLY, SPIFFS_WR_ONLY, SPIFFS_RDWR, SPIFFS_DIRECT. SPIFFS_CREAT will have no effect in this case.
- `mode`: Ignored, for posix compliance.

*spiffs_file_t* **spiffs_open_by_page**(**struct** *spiffs_t* *self_p, spiffs_page_ix_t *page_ix*, *spiffs_flags_t* flags, *spiffs_mode_t* mode)

Opens a file by given page index.

Optimization purposes, opens a file by directly pointing to the page index in the spi flash. If the page index does not point to a file header SPIFFS_ERR_NOT_A_FILE is returned.

**Parameters**

- `self_p`: The file system struct.
- `page_ix`: The page index.
- `flags`: The flags for the open command, can be combinations of SPIFFS_APPEND, SPIFFS_TRUNC, SPIFFS_CREAT, SPIFFS_RD_ONLY, SPIFFS_WR_ONLY, SPIFFS_RDWR, SPIFFS_DIRECT. SPIFFS_CREAT will have no effect in this case.
- `mode`: Ignored, for posix compliance.

int32_t **spiffs_read**(**struct** *spiffs_t* *self_p, *spiffs_file_t* fh, void *buf_p, int32_t *len*)

Reads from given filehandle.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle.
- `buf_p`: Where to put read data.
- `len`: How much to read.

int32_t **spiffs_write**(**struct** *spiffs_t* *self_p, *spiffs_file_t* fh, void *buf_p, int32_t len)
Writes to given filehandle.

**Return** Number of bytes written, or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle.
- `buf_p`: The data to write.
- `len`: How much to write.

int32_t **spiffs_lseek**(**struct** *spiffs_t* *self_p, *spiffs_file_t* fh, int32_t offs, int whence)
Moves the read/write file offset. Resulting offset is returned or negative if error.

lseek(fs, fd, 0, SPIFFS_SEEK_CUR) will thus return current offset.

If SPIFFS_SEEK_CUR, the file offset shall be set to its current location plus offset.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle.
- `offs`: How much/where to move the offset.
- `whence`: If SPIFFS_SEEK_SET, the file offset shall be set to offset bytes.

If SPIFFS_SEEK_END, the file offset shall be set to the size of the file plus offse, which should be negative.

**Return** zero(0) or negative error code.

int32_t **spiffs_remove**(**struct** *spiffs_t* *self_p, **const** char *path_p)
Removes a file by path.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `path_p`: The path of the file to remove.

int32_t **spiffs_fremove**(**struct** *spiffs_t* *self_p, *spiffs_file_t* fh)
Removes a file by filehandle.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle of the file to remove.

int32_t **spiffs_stat**(**struct** *spiffs_t* *self_p, **const** char *path_p, **struct** *spiffs_stat_t* *stat_p)
Gets file status by path.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `path_p`: The path of the file to stat.
- `stat_p`: The stat struct to populate.

int32_t **spiffs_fstat**(**struct** *spiffs_t* *self_p, *spiffs_file_t* fh, **struct** *spiffs_stat_t* *stat_p)
Gets file status by filehandle.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle of the file to stat.
- `stat_p`: The stat struct to populate.

int32_t **spiffs_fflush**(**struct** *spiffs_t* *self_p, *spiffs_file_t* fh)
Flushes all pending write operations from cache for given file.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle of the file to flush.

int32_t **spiffs_close**(**struct** *spiffs_t* *self_p, *spiffs_file_t* fh)
Closes a filehandle. If there are pending write operations, these are finalized before closing.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle of the file to close.

int32_t **spiffs_rename**(**struct** *spiffs_t* *self_p, **const** char *old_path_p, **const** char *new_path_p)
Renames a file.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `old_path_p`: Path of file to rename.

- `new_path_p`: New path of file.

int32_t **spiffs_errno**(**struct** *spiffs_t *self_p*)
Returns last error of last file operation.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.

void **spiffs_clearerr**(**struct** *spiffs_t *self_p*)
Clears last error.

**Return** void.

**Parameters**

- `self_p`: The file system struct.

**struct** *spiffs_dir_t* ***spiffs_opendir**(**struct** *spiffs_t *self_p*, **const** char *name_p*, **struct** *spiffs_dir_t *dir_p*)
Opens a directory stream corresponding to the given name. The stream is positioned at the first entry in the directory. On hydrogen builds the name argument is ignored as hydrogen builds always correspond to a flat file structure - no directories.

**Parameters**

- `self_p`: The file system struct.
- `name_p`: The name of the directory.
- `dir_p`: Pointer the directory stream to be populated.

int32_t **spiffs_closedir**(**struct** *spiffs_dir_t *dir_p*)
Closes a directory stream

**Return** zero(0) or negative error code.

**Parameters**

- `dir_p`: The directory stream to close.

**struct** *spiffs_dirent_t* ***spiffs_readdir**(**struct** *spiffs_dir_t *dir_p*, **struct** *spiffs_dirent_t *ent_p*)
Reads a directory into given spifs_dirent struct.

**Return** NULL if error or end of stream, else given dirent is returned.

**Parameters**

- `dir_p`: Pointer to the directory stream.
- `ent_p`: The dirent struct to be populated.

int32_t **spiffs_check**(**struct** *spiffs_t *self_p*)
Runs a consistency check on given filesystem.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.

int32_t **spiffs_info**(**struct** *spiffs_t* *self_p, uint32_t *total_p, uint32_t *used_p)

Returns number of total bytes available and number of used bytes. This is an estimation, and depends on if there a many files with little data or few files with much data.

NB: If used number of bytes exceeds total bytes, a SPIFFS_check should run. This indicates a power loss in midst of things. In worst case (repeated powerlosses in mending or gc) you might have to delete some files.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `total_p`: Total number of bytes in filesystem.
- `used_p`: Used number of bytes in filesystem.

int32_t **spiffs_format**(**struct** *spiffs_t* *self_p)

Formats the entire file system. All data will be lost. The filesystem must not be mounted when calling this.

NB: formatting is awkward. Due to backwards compatibility, SPIFFS_mount MUST be called prior to formatting in order to configure the filesystem. If SPIFFS_mount succeeds, SPIFFS_unmount must be called before calling SPIFFS_format. If SPIFFS_mount fails, SPIFFS_format can be called directly without calling SPIFFS_unmount first.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.

uint8_t **spiffs_mounted**(**struct** *spiffs_t* *self_p)

Returns nonzero if spiffs is mounted, or zero if unmounted.

**Parameters**

- `self_p`: The file system struct.

int32_t **spiffs_gc_quick**(**struct** *spiffs_t* *self_p, uint16_t *max_free_pages*)

Tries to find a block where most or all pages are deleted, and erase that block if found. Does not care for wear levelling. Will not move pages around.

If parameter max_free_pages are set to 0, only blocks with only deleted pages will be selected.

NB: the garbage collector is automatically called when spiffs needs free pages. The reason for this function is to give possibility to do background tidying when user knows the system is idle.

Use with care.

Setting max_free_pages to anything larger than zero will eventually wear flash more as a block containing free pages can be erased.

Will set err_no to SPIFFS_OK if a block was found and erased, SPIFFS_ERR_NO_DELETED_BLOCK if no matching block was found, or other error.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.

- `max_free_pages`: maximum number allowed free pages in block.

int32_t **spiffs_gc**(**struct** *spiffs_t *self_p*, uint32_t *size*)
Will try to make room for given amount of bytes in the filesystem by moving pages and erasing blocks. If it is physically impossible, err_no will be set to SPIFFS_ERR_FULL. If there already is this amount (or more) of free space, SPIFFS_gc will silently return. It is recommended to call SPIFFS_info before invoking this method in order to determine what amount of bytes to give.

NB: the garbage collector is automatically called when spiffs needs free pages. The reason for this function is to give possibility to do background tidying when user knows the system is idle.

Use with care.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `size`: Amount of bytes that should be freed.

int32_t **spiffs_eof**(**struct** *spiffs_t *self_p*, *spiffs_file_t fh*)
Check if EOF reached.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle of the file to check.

int32_t **spiffs_tell**(**struct** *spiffs_t *self_p*, *spiffs_file_t fh*)
Get position in file.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `fh`: The filehandle of the file to check.

int32_t **spiffs_set_file_callback_func**(**struct** *spiffs_t *self_p*, *spiffs_file_callback_t cb_func*)
Registers a callback function that keeps track on operations on file headers. Do note, that this callback is called from within internal spiffs mechanisms. Any operations on the actual file system being callbacked from in this callback will mess things up for sure - do not do this. This can be used to track where files are and move around during garbage collection, which in turn can be used to build location tables in ram. Used in conjuction with SPIFFS_open_by_page this may improve performance when opening a lot of files. Must be invoked after mount.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The file system struct.
- `cb_func`: The callback on file operations.

**struct spiffs_config_t**
*#include* <*spiffs.h*> Spiffs spi configuration struct.

### Public Members

*spiffs_read_cb_t* **hal_read_f**
Physical read function.

*spiffs_write_cb_t* **hal_write_f**
Physical write function.

*spiffs_erase_cb_t* **hal_erase_f**
Physical erase function.

uint32_t **phys_size**
Physical size of the spi flash.

uint32_t **phys_addr**
Physical offset in spi flash used for spiffs, must be on block boundary.

uint32_t **phys_erase_block**
Physical size when erasing a block.

uint32_t **log_block_size**
Logical size of a block, must be on physical block size boundary and must never be less than a physical block.

uint32_t **log_page_size**
Logical size of a page, must be at least log_block_size /

1.

**struct spiffs_t**

### Public Members

**struct** *spiffs_config_t* **cfg**
File system configuration.

uint32_t **block_count**
Number of logical blocks.

spiffs_block_ix_t **free_cursor_block_ix**
Cursor for free blocks, block index.

int **free_cursor_obj_lu_entry**
Cursor for free blocks, entry index.

spiffs_block_ix_t **cursor_block_ix**
Cursor when searching, block index.

int **cursor_obj_lu_entry**
Cursor when searching, entry index.

uint8_t ***lu_work**
Primary work buffer, size of a logical page.

uint8_t ***work**
Secondary work buffer, size of a logical page.

uint8_t ***fd_space**
File descriptor memory area.

uint32_t **fd_count**
Available file descriptors.

int32_t **err_code**
Last error.

uint32_t **free_blocks**
Current number of free blocks.

uint32_t **stats_p_allocated**
Current number of busy pages.

uint32_t **stats_p_deleted**
Current number of deleted pages.

uint8_t **cleaning**
Flag indicating that garbage collector is cleaning.

spiffs_obj_id_t **max_erase_count**
Max erase count amongst all blocks.

*spiffs_check_callback_t* **check_cb_f**
Check callback function.

*spiffs_file_callback_t* **file_cb_f**
File callback function.

uint8_t **mounted**
Mounted flag.

void ***user_data**
User data.

uint32_t **config_magic**
Config magic.

**struct spiffs_stat_t**
*#include* <*spiffs.h*> Spiffs file status struct.

### Public Members

spiffs_obj_id_t **obj_id**

uint32_t **size**

*spiffs_obj_type_t* **type**

spiffs_page_ix_t **pix**

uint8_t **name**[**SPIFFS_OBJ_NAME_LEN**]

**struct spiffs_dirent_t**

### Public Members

spiffs_obj_id_t **obj_id**

uint8_t **name**[**SPIFFS_OBJ_NAME_LEN**]

*spiffs_obj_type_t* **type**

uint32_t **size**

spiffs_page_ix_t **pix**

**struct spiffs_dir_t**

#### Public Members

**struct** *spiffs_t* ***fs**

spiffs_block_ix_t **block**

int **entry**

### 1.6.5 inet

The inet package on Github.

Modules:

#### `http_server` — HTTP server

A HTTP server serves HTTP client requests, typically from a web browser.

A HTTP server can be wrapped in SSL, a secutiry layer, to create a HTTPS server.

---

Source code: src/inet/http_server.h, src/inet/http_server.c

Test code: tst/inet/http_server/main.c

Test coverage: src/inet/http_server.c

Example code: examples/http_server/main.c, examples/https_server/main.c

---

#### Typedefs

**typedef** int (***http_server_route_callback_t**)(**struct** *http_server_connection_t* *connection_p, **struct** *http_server_request_t* *request_p)

#### Enums

**enum http_server_request_action_t**

Request action types.

*Values:*

**http_server_request_action_get_t** = 0

**http_server_request_action_post_t** = 1

**enum http_server_content_type_t**

Content type.

*Values:*

**http_server_content_type_text_plain_t** = 0

**http_server_content_type_text_html_t** = 1

**enum http_server_response_code_t**
Response codes.

*Values:*

**http_server_response_code_200_ok_t** = 200

**http_server_response_code_400_bad_request_t** = 400

**http_server_response_code_401_unauthorized_t** = 401

**http_server_response_code_404_not_found_t** = 404

**enum http_server_connection_state_t**
Connection state.

*Values:*

**http_server_connection_state_free_t** = 0

**http_server_connection_state_allocated_t**

### Functions

int **http_server_init**(**struct** *http_server_t* *self_p, **struct** *http_server_listener_t* *listener_p, **struct** *http_server_connection_t* *connections_p, **const** char *root_path_p, **const struct** *http_server_route_t* *routes_p, *http_server_route_callback_t* on_no_route)
Initialize given http server with given root path and maximum number of clients.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http server to initialize.
- `listener_p`: Listener.
- `connections_p`: A NULL terminated list of connections.
- `root_path_p`: Working directory for the connection threads.
- `routes_p`: An array of routes.
- `on_no_route`: Callback called for all requests without a matching route in route_p.

int **http_server_wrap_ssl**(**struct** *http_server_t* *self_p, **struct** *ssl_context_t* *context_p)
Wrap given HTTP server in SSL, to make it secure.

This function must be called after `http_server_init()` and before `http_server_start()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http server to wrap in SSL.
- `context_p`: SSL context to wrap the server in.

int **http_server_start**(**struct** *http_server_t* *self_p)
Start given HTTP server.

Spawn the threads and start listening for connections.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http server.

int **http_server_stop**(**struct** *http_server_t* **self_p*)

Stop given HTTP server.

Closes the listener and all open connections, and then kills the threads.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http server.

int **http_server_response_write**(**struct** *http_server_connection_t* **connection_p*, **struct** *http_server_request_t* **request_p*, **struct** *http_server_response_t* **response_p*)

Write given HTTP response to given connected client. This function should only be called from the route callbacks to respond to given request.

**Return** zero(0) or negative error code.

**Parameters**

- `connection_p`: Current connection.
- `request_p`: Current request.
- `response_p`: Current response. If `buf_p` in the response to NULL this function will only write the HTTP header, including the size, to the socket. After this function returns write the payload by calling `socket_write()`.

**struct http_server_request_t**

*#include <http_server.h>* HTTP request.

### Public Members

*http_server_request_action_t* **action**

char **path**[64]

int **present**

char **value**[20]

**struct http_server_request_t::@56::@57 http_server_request_t::sec_websocket_key**

**struct http_server_request_t::@56::@58 http_server_request_t::content_type**

long **value**

**struct http_server_request_t::@56::@59 http_server_request_t::content_length**

**struct http_server_request_t::@56::@60 http_server_request_t::authorization**

**struct http_server_request_t::@56::@61 http_server_request_t::expect**

**struct http_server_request_t::@56 http_server_request_t::headers**

**struct http_server_response_t**
*#include <http_server.h>* HTTP response.

### Public Members

int **type**

*http_server_response_code_t* **code**

**const** char ***buf_p**

size_t **size**

**struct http_server_response_t::@62 http_server_response_t::content**

**struct http_server_listener_t**

### Public Members

**const** char ***address_p**

int **port**

**const** char ***name_p**

void ***buf_p**

size_t **size**

**struct http_server_listener_t::@63::@64 http_server_listener_t::stack**

**struct** *thrd_t* ***id_p**

**struct http_server_listener_t::@63 http_server_listener_t::thrd**

**struct** *socket_t* **socket**

**struct http_server_connection_t**

### Public Members

*http_server_connection_state_t* **state**

**const** char ***name_p**

void ***buf_p**

size_t **size**

**struct http_server_connection_t::@65::@66 http_server_connection_t::stack**

**struct** *thrd_t* ***id_p**

**struct http_server_connection_t::@65 http_server_connection_t::thrd**

**struct** *http_server_t* ***self_p**

**struct** *socket_t* **socket**

void ***chan_p**

**struct** *event_t* **events**

**struct http_server_route_t**
*#include <http_server.h>* Call given callback for given path.

### Public Members

**const** char ***path_p**

*http_server_route_callback_t* **callback**

**struct http_server_t**

### Public Members

**const** char ***root_path_p**

**const struct** *http_server_route_t* ***routes_p**

*http_server_route_callback_t* **on_no_route**

**struct** *http_server_listener_t* ***listener_p**

**struct** *http_server_connection_t* ***connections_p**

**struct** *ssl_context_t* ***ssl_context_p**

**struct** *event_t* **events**

## http_websocket_client — HTTP websocket client

Source code: src/inet/http_websocket_client.h, src/inet/http_websocket_client.c

Test code: tst/inet/http_websocket_client/main.c

Test coverage: src/inet/http_websocket_client.c

---

### Functions

int **http_websocket_client_init**(**struct** *http_websocket_client_t* **self_p*, **const** char **server_p*, int *port*, **const** char **path_p*)
Initialize given http.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http to initialize.
- `server_p`: Server hostname to connect to.
- `port`: Port to connect to.
- `path_p`: Path.

int **http_websocket_client_connect**(**struct** *http_websocket_client_t* **self_p*)
Connect given http to the server.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http to connect.

int **http_websocket_client_disconnect**(**struct** *http_websocket_client_t* *self_p)

Disconnect given http from the server.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http to connect.

ssize_t **http_websocket_client_read**(**struct** *http_websocket_client_t* *self_p, void *buf_p, size_t size)

Read from given http.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Http to read from.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read..

ssize_t **http_websocket_client_write**(**struct** *http_websocket_client_t* *self_p, int type, **const** void *buf_p, uint32_t size)

Write given data to given http.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Http to write to.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

**struct http_websocket_client_t**

### Public Members

**struct** *socket_t* **socket**

**const** char ***host_p**

int **port**

**struct http_websocket_client_t::@67 http_websocket_client_t::server**

size_t **left**

**struct http_websocket_client_t::@68 http_websocket_client_t::frame**

**const** char ***path_p**

### `http_websocket_server` — HTTP websocket server

Source code: src/inet/http_websocket_server.h, src/inet/http_websocket_server.c

Test code: tst/inet/http_websocket_server/main.c

Test coverage: src/inet/http_websocket_server.c

---

## Functions

int **http_websocket_server_init**(**struct** *http_websocket_server_t* **self_p*, **struct** *socket_t* **socket_p*)

Initialize given websocket server. The server uses the http module interface to communicate with the client.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Http to initialize.
- `socket_p`: Connected socket.

int **http_websocket_server_handshake**(**struct** *http_websocket_server_t* **self_p*, **struct** *http_server_request_t* **request_p*)

Read the handshake request from the client and send the handshake response.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Websocket server.
- `request_p`: Read handshake request.

ssize_t **http_websocket_server_read**(**struct** *http_websocket_server_t* **self_p*, int **type_p*, void **buf_p*, size_t *size*)

Read a message from given websocket.

**Return** Number of bytes read or negative error code.

**Parameters**

- `self_p`: Websocket to read from.
- `type_p`: Read message type.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read. Longer messages will be truncated and the leftover data dropped.

ssize_t **http_websocket_server_write**(**struct** *http_websocket_server_t* **self_p*, int *type*, **const** void **buf_p*, uint32_t *size*)

Write given message to given websocket.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Websocket to write to.

- `type`: One of `HTTP_TYPE_TEXT` and `HTTP_TYPE_BINARY`.
- `buf_p`: Buffer to write.
- `size`: Number of bytes to write.

**struct http_websocket_server_t**

#### Public Members

**struct** *socket_t* ***socket_p**

### inet — Internet utilities

Source code: src/inet/inet.h, src/inet/inet.c

Test code: tst/inet/inet/inet.c

Test coverage: src/inet/inet.c

---

#### Functions

int **inet_module_init**(void)

Initialize the inet module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **inet_aton**(**const** char **src_p*, **struct** *inet_ip_addr_t* **dst_p*)

Convert the Internet host address src_p from the IPv4 numbers-and-dots notation into binary form (in network byte order) and stores it in the structure that dst_p points to.

The address supplied in src_p can have one of the following forms:

- a.b.c.d Each of the four numeric parts specifies a byte of the address; the bytes are assigned in left-to-right order to produce the binary address.

**Return** zero(0) or negative error code.

**Parameters**

- `src_p`: Address a.b.c.d to convert into a number.
- `dst_p`: Converted address.

char ***inet_ntoa**(**const struct** *inet_ip_addr_t* **src_p*, char **dst_p*)

Convert the Internet host src_p from the IPv4 binary form (in network byte order) to numbers-and-dots notation and stores it in the structure that dst_p points to.

**Return** Converted address pointer or NULL on failure.

**Parameters**

- `src_p`: Address to convert into a string.

- `dst_p`: Converted address as a string.

uint16_t **inet_checksum**(void *_buf_p_, size_t _size_)
Calculate the internet checksum of given buffer.

**Return** Calculated checksum.

**Parameters**

- `buf_p`: Buffer to calculate the chechsum of.
- `size`: Size of the buffer.

**struct inet_ip_addr_t**

#### Public Members

uint32_t **number**
IPv4 address.

**struct inet_addr_t**

#### Public Members

**struct** *inet_ip_addr_t* **ip**
IPv4 address.

uint16_t **port**
Port.

**struct inet_if_ip_info_t**
*#include* <*inet.h*> Interface IP information.

#### Public Members

**struct** *inet_ip_addr_t* **address**

**struct** *inet_ip_addr_t* **netmask**

**struct** *inet_ip_addr_t* **gateway**

### isotp — ISO-TP

Source code: src/inet/isotp.h, src/inet/isotp.c

Test code: tst/inet/isotp/main.c

---

#### Defines

**ISOTP_FLAGS_NO_FLOW_CONTROL**

### Functions

int **isotp_init**(**struct** *isotp_t* *self_p, uint8_t *message_p, size_t size, int flags)

Initialize given ISO-TP object. An object can *either* be used to transmit or receive an ISO-TP message. Once `isotp_input()` or `isotp_output()` returns a positive value the message transmission is completed.

An object can only be used to transmit one message. Initialize a new object to transmit another message.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object to initialize.
- `message_p`: ISO-TP message to transmit, or a reception buffer for an incoming message.
- `size`: Size of the message buffer in bytes.
- `flags`: Configuration flags.

ssize_t **isotp_input**(**struct** *isotp_t* *self_p, **const** uint8_t *buf_p, size_t size)

Input a CAN frame into given ISO-TP object. Always call isotp_output() after this function returns zero(0) to check if there are frames to transmit.

For an ISO-TP object that transmits a message this function always returns zero(0) or negative error code.

**Return** Once a complete ISO-TP message has been received the size of the message is returned. Meanwhile, zero(0) is returned if the frame was expected. A negative error code is returned if the frame was unexpected or invalid.

**Parameters**

- `self_p`: Initialized ISO-TP object.
- `buf_p`: Input data.
- `size`: Data buffer length is bytes.

ssize_t **isotp_output**(**struct** *isotp_t* *self_p, uint8_t *buf_p, size_t *size_p)

Check if there is data to be transmitted. The caller must transmit all frames this function creates.

For an ISO-TP object that receives a message this function always returns zero(0) or negative error code.

**Return** Once a complete ISO-TP message has been transmitted the size of the message is returned. Meanwhile, zero(0) or negative error code is returned.

**Parameters**

- `self_p`: Initialized ISO-TP object.
- `buf_p`: Output data to be transmitted to the peer. The size of this buffer must be at least eight bytes.
- `size_p`: Number of bytes to be transmitted.

**struct isotp_t**

#### Public Members

uint8_t ***message_p**

size_t **size**

int **state**

int **flags**

size_t **offset**

int **next_index**

**struct isotp_t::@69 isotp_t::message**

### mqtt_client — MQTT client

MQTT is a publish-subscribe-based lightweight messaging protocol.

The driver works by running the processing code in a thread which communicate with the MQTT broker on one side using channels, and the application on the other side using queues.

This means the application has to set up appropriate channels, which is already ready to communicate with the MQTT server, e.g. using TCP, and the thread running the MQTT client.

### MQTT Notes

MQTT requires the client to send packets regularly, otherwise the broker will disconnect the client. Any packet from the client to the broker will fulfill the requirement, so a regular stream of publish messages will work or sending MQTT Ping packets as needed. The keep alive interval is set by the client on connect, and can be changed from the default in the mqtt_conn_options_t.

---

**Note:** The current client does not gracefully handle the underlying channel (e.g. TCP connection) to the broker disconnecting, and requires complete restart of the MQTT client to recover.

---

### Basic MQTT client usage

Basic example of initializing MQTT over TCP (error checking left out for brevity).

```
static size_t on_publish(struct mqtt_client_t *client_p,
                         const char *topic_p,
                         void *chin_p,
                         size_t size)
{
    uint8_t buf[32];

    chan_read(chin_p, buf, size);
    buf[size] = '\0';
    std_printf(OSTR("on_publish: %s\r\n"), &buf[0]);

    return (0);
}
```

```
struct inet_addr_t remote_host_address;

inet_aton("127.0.0.1", &remote_host_address.ip);
remote_host_address.port = 1883;
socket_open_tcp(&server_sock);
socket_connect(&server_sock, &remote_host_address);
```

(continues on next page)

(continued from previous page)

```
mqtt_client_init(&client,
                 "mqtt_client",
                 NULL,
                 &server_sock,
                 &server_sock,
                 on_publish,
                 NULL);

thrd_spawn(mqtt_client_main,
           &client,
           0,
           stack,
           sizeof(stack));

mqtt_client_connect(&client);
```

Source code: src/inet/mqtt_client.h, src/inet/mqtt_client.c

Test code: tst/inet/mqtt_client/main.c

Test coverage: src/inet/mqtt_client.c

Example code: examples/mqtt_client/main.c

---

### Defines

**DEFAULT_KEEP_ALIVE_S**
Default MQTT keep alive interval in seconds.

### Typedefs

**typedef** size_t (***mqtt_on_publish_t**)(**struct** *mqtt_client_t* *client_p, **const** char *topic_p, void *chin_p, size_t size)
Prototype of the on-publish callback function.

**Return** Number of bytes read from the input channel.

**Parameters**

- `client_p`: The client.
- `topic_p`: The received topic.
- `chin_p`: The channel to read the value from.
- `size`: Number of bytes of the value to read from chin_p.

**typedef** int (***mqtt_on_error_t**)(**struct** *mqtt_client_t* *client_p, int error)
Prototype of the on-error callback function.

**Return** zero(0) or nagative error code.

**Parameters**

- `client_p`: The client.

- `error`: The number of error that occured.

### Enums

**enum mqtt_client_state_t**
Client states.

*Values:*

**mqtt_client_state_disconnected_t**

**mqtt_client_state_connected_t**

**mqtt_client_state_connecting_t**

**enum mqtt_qos_t**
Quality of Service.

*Values:*

**mqtt_qos_0_t** = 0

**mqtt_qos_1_t** = 1

**mqtt_qos_2_t** = 2

### Functions

int **mqtt_client_init**(**struct** *mqtt_client_t* *self_p, **const** char *name_p, **struct** *log_object_t* *log_object_p, void *chout_p, void *chin_p, *mqtt_on_publish_t* on_publish, *mqtt_on_error_t* on_error)
Initialize given MQTT client.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: MQTT client.
- `name_p`: Name of the thread.
- `log_object_p`: Log object.
- `chout_p`: Output channel for client to server packets.
- `chin_p`: Input channel for server to client packets.
- `on_publish`: On-publish callback function. Called when the server publishes a message.
- `on_error`: On-error callback function. Called when an error occurs. If NULL, a default handler is used.

void *mqtt_client_main(void *arg_p)
MQTT client thread.

**Return** Never returns.

**Parameters**

- `arg_p`: MQTT client.

int **mqtt_client_connect**(**struct** *mqtt_client_t *self_p*, **struct** *mqtt_conn_options_t *options_p*)
Establish a connection to the server.

**Warning** If options_p is set, all members of the struct not explicitly used, must be set to zero. It is suggested to do this by calling memset(options_p, 0, sizeof(*options_p)); before setting needed variables.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: MQTT client.
- `options_p`: MQTT connection options. May be NULL. Pointer only need to be valid for the duration of the function call.

int **mqtt_client_disconnect**(**struct** *mqtt_client_t *self_p*)
Disconnect from the server.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: MQTT client.

int **mqtt_client_ping**(**struct** *mqtt_client_t *self_p*)
Send a ping request to the server (broker) and wait for the ping response.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: MQTT client.

int **mqtt_client_publish**(**struct** *mqtt_client_t *self_p*, **struct** *mqtt_application_message_t *message_p*)
Publish given topic.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: MQTT client.
- `topic_p`: Topic.
- `payload_p`: Payload to publish. May be NULL.
- `payload_size`: Number of bytes in the payload.

int **mqtt_client_subscribe**(**struct** *mqtt_client_t *self_p*, **struct** *mqtt_application_message_t *message_p*)
Subscribe to given message.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: MQTT client.
- `message_p`: The message to subscribe to. The payload part of the message is not used. The topic may use wildcards, given that the server supports it.

int **mqtt_client_unsubscribe**(**struct** *mqtt_client_t* *self_p, **struct** *mqtt_application_message_t* *message_p)

Unsubscribe from given message.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: MQTT client.
- `message_p`: The message to unsubscribe from. Only the topic in the message is used.

**struct mqtt_string_t**

*#include <mqtt_client.h>* An MQTT style length string.

### Public Members

**const** void ***buf_p**

size_t **size**

**struct mqtt_client_t**

*#include <mqtt_client.h>* MQTT client.

### Public Members

**const** char ***name_p**

**struct** *log_object_t* ***log_object_p**

int **state**

int **type**

void ***data_p**

**struct mqtt_client_t::@70 mqtt_client_t::message**

void ***out_p**

void ***in_p**

**struct mqtt_client_t::@71 mqtt_client_t::transport**

**struct** *queue_t* **out**

**struct** *queue_t* **in**

**struct mqtt_client_t::@72 mqtt_client_t::control**

*mqtt_on_publish_t* **on_publish**

*mqtt_on_error_t* **on_error**

**struct mqtt_application_message_t**

*#include <mqtt_client.h>* MQTT application message.

#### Public Members

**struct** *mqtt_string_t* **topic**

**struct** *mqtt_string_t* **payload**

*mqtt_qos_t* **qos**

**struct mqtt_conn_options_t**

*#include <mqtt_client.h>* MQTT Connection options.

#### Public Members

**struct** *mqtt_string_t* **client_id**
Should be 1-23 [0-9a-zA-Z] characters as per [MQTT-3.1.3-5].

**struct** *mqtt_application_message_t* **will**
Optional Last Will and Testament to be sent on unclean disconnect.

**struct** *mqtt_string_t* **user_name**
Optional user name for broker authentication.

**struct** *mqtt_string_t* **password**
Optional password for broker authentication.

int **keep_alive_s**
Keep alive interval in seconds.

### network_interface — Network interface

The network interface module has a list of all network interfaces and their states.

Network interface modules:

### network_interface_slip — Serial Link Internet Protocol

Serial Line Internet Protocol (SLIP) is a link layer internet protocol used to transfer TCP/IP packets over a point-to-point serial line.

It is documented in RFC 1055.

---

Source code: src/inet/network_interface/slip.h

Example code: examples/inet/slip/main.c

---

#### Defines

**NETWORK_INTERFACE_SLIP_FRAME_SIZE_MAX**

### Enums

**enum network_interface_slip_state_t**

*Values:*

**NETWORK_INTERFACE_SLIP_STATE_NORMAL** = 0

**NETWORK_INTERFACE_SLIP_STATE_ESCAPE**

### Functions

int **network_interface_slip_module_init**(void)

Initialize the slip module.

**Return** zero(0) or negative error code.

int **network_interface_slip_init**(**struct** *network_interface_slip_t* *self_p, **struct** *inet_ip_addr_t* *ipaddr_p, **struct** *inet_ip_addr_t* *netmask_p, **struct** *inet_ip_addr_t* *gateway_p, void *chout_p)

Initialize given slip network interface with given configuration and output channel.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Slip to initialize.
- `ipaddr_p`: Network interface IP address.
- `netmask_p`: Network interface netmask.
- `gateway_p`: Network interface gateway.
- `chout_p`: Output channel.

int **network_interface_slip_input**(**struct** *network_interface_slip_t* *self_p, uint8_t *data*)

Input a byte into the SLIP IP stack. Normally a user thread reads one byte at a time from the UART and calls this functions with the read byte as argument.

**Return** Number of bytes written to the input frame or negative error code.

**Parameters**

- `self_p`: Slip to initialize.
- `data`: Byte to input into the stack.

**struct network_interface_slip_t**

#### Public Members

*network_interface_slip_state_t* **state**

**struct** pbuf ***pbuf_p**

uint8_t ***buf_p**

size_t **size**

**struct network_interface_slip_t::@73 network_interface_slip_t::frame**

void ***chout_p**

**struct** *network_interface_t* **network_interface**

### network_interface_wifi — WiFi network interface

WiFi network interface driver modules:

### network_interface_driver_esp — ESP WiFi network interface driver

---

Source code: src/inet/network_interface/driver/esp.h, src/inet/network_interface/driver/esp.c

Test code: tst/inet/network_interface/wifi_esp/main.c

---

#### Variables

**struct** *network_interface_wifi_driver_t* **network_interface_wifi_driver_esp_station**
Esressif WiFi Station driver callbacks. To be used as driver in the wifi network interface.

**struct** *network_interface_wifi_driver_t* **network_interface_wifi_driver_esp_softap**
Esressif WiFi SoftAP driver callbacks. To be used as driver in the wifi network interface.

---

Source code: src/inet/network_interface/wifi.h, src/inet/network_interface/wifi.c

Test code: tst/inet/network_interface/wifi_esp/main.c

---

#### Functions

int **network_interface_wifi_module_init**(void)
Initialize the WiFi network interface module.

**Return** zero(0) or negative error code.

int **network_interface_wifi_init**(**struct** *network_interface_wifi_t* ***self_p**, **const** char ***name_p**, **struct** *network_interface_wifi_driver_t* ***driver_p**, void ***arg_p**, **const** char ***ssid_p**, **const** char ***password_p**)
Initialize given WiFi network interface with given configuration.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: The WiFi network interface to initialize.
- `name_p`: Name to assign the to interface.

- `driver_p`: Driver virtualization callbacks to use.
- `arg_p`: Argument passed to the driver callbacks. In case of ESP chips and WiFi station mode - compound literal of uint8_t[6] specifying the access point MAC.
- `ssid_p`: Access Point SSID.
- `password_p`: Access Point password.

int **network_interface_wifi_start**(**struct** *network_interface_wifi_t *self_p*)
Start given WiFi network interface.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: WiFi network interface to start.

int **network_interface_wifi_stop**(**struct** *network_interface_wifi_t *self_p*)
Stop given WiFi network interface.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: WiFi network interface to stop.

int **network_interface_wifi_is_up**(**struct** *network_interface_wifi_t *self_p*)
Get the connection status of given network interface.

**Return** true(1) if the network interface is up, false(0) is it is down, and otherwise negative error code.

**Parameters**

- `self_p`: Network interface to get the connection status of.

int **network_interface_wifi_set_ip_info**(**struct** *network_interface_wifi_t *self_p*, **const struct** *inet_if_ip_info_t *info_p*)
Set the ip address, netmask and gateway of given network interface.

**Return** zero(0) if the interface has valid IP information, otherwise negative error code.

**Parameters**

- `self_p`: Network interface.
- `info_p`: Interface IP information to set.

int **network_interface_wifi_get_ip_info**(**struct** *network_interface_wifi_t *self_p*, **struct** *inet_if_ip_info_t *info_p*)
Get the ip address, netmask and gateway of given network interface.

**Return** zero(0) if the interface has valid IP information, otherwise negative error code.

**Parameters**

- `self_p`: Network interface.
- `info_p`: Interface IP information. Only valid if this function returns zero(0).

**struct network_interface_wifi_t**
*#include* <*wifi.h*> A WiFi network interface.

#### Public Members

**struct** *network_interface_t* **network_interface**

**struct** *network_interface_wifi_driver_t* ***driver_p**

void ***arg_p**

**const** char ***ssid_p**

**const** char ***password_p**

**const struct** *inet_if_ip_info_t* ***info_p**

**struct network_interface_wifi_driver_t**

*#include* <*wifi.h*> Driver virtualization callbacks. See the `driver/` subfolder for available drivers.

#### Public Members

int (***init**)(void *arg_p)

int (***start**)(void *arg_p, **const** char *ssid_p, **const** char *password_p, **const struct** *inet_if_ip_info_t* *info_p)

int (***stop**)(void *arg_p)

int (***is_up**)(void *arg_p)

int (***set_ip_info**)(void *arg_p, **const struct** *inet_if_ip_info_t* *info_p)

int (***get_ip_info**)(void *arg_p, **struct** *inet_if_ip_info_t* *info_p)

### Debug file system commands

One debug file system command is available, located in the directory `inet/network_interface/`.

| Command | Description |
|---|---|
| `list` | Print a list of all registered network interfaces. |

Example output from the shell:

```
$ inet/network_interface/list
NAME            STATE  ADDRESS            TX BYTES    RX BYTES
esp-wlan-ap     up     192.168.4.1               -           -
esp-wlan-sta    up     192.168.0.5               -           -
OK
```

---

Source code: src/inet/network_interface.h, src/inet/network_interface.c

Test coverage: src/inet/network_interface.c

---

### Typedefs

**typedef** int (***network_interface_start_t**)(**struct** *network_interface_t* *netif_p)

**typedef** int (***network_interface_stop_t**)(**struct** *network_interface_t* *netif_p)

**typedef** int (***network_interface_is_up_t**)(**struct** *network_interface_t* *netif_p)

**typedef** int (***network_interface_set_ip_info_t**)(**struct** *network_interface_t* *netif_p, **const struct** *inet_if_ip_info_t* *info_p)

**typedef** int (***network_interface_get_ip_info_t**)(**struct** *network_interface_t* *netif_p, **struct** *inet_if_ip_info_t* *info_p)

### Functions

int **network_interface_module_init**(void)

Initialize the network interface module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **network_interface_add**(**struct** *network_interface_t* **netif_p*)

Add given network interface to the global list of network interfaces. Call `network_interface_start()` to enable the interface.

**Return** zero(0) or negative error code.

**Parameters**

- `netif_p`: Network interface to register.

int **network_interface_start**(**struct** *network_interface_t* **netif_p*)

Start given network interface. Enables the interface in the IP stack to allow packets to be sent and received. If the interface is a WiFi station interface it will try initiate the connection to its configured access point. Use `network_interface_is_up()` to check if the interface is connected to its access point.

**Return** zero(0) or negative error code.

**Parameters**

- `netif_p`: Network interface to start.

int **network_interface_stop**(**struct** *network_interface_t* **netif_p*)

Stop given network interface. Disconnects from any WiFi access points and disables the interface in the IP stack. No packets can be sent or received on this interface after this function is called.

**Return** zero(0) or negative error code.

**Parameters**

- `netif_p`: Network interface to stop.

int **network_interface_is_up**(**struct** *network_interface_t* **netif_p*)

Get the connection status of given network interface. PAckets can only be sent and received when the interface in up.

**Return** true(1) if the network interface is up, false(0) is it is down, and otherwise negative error code.

**Parameters**

- `netif_p`: Network interface to get the connection status of.

**struct** *network_interface_t* ***network_interface_get_by_name**(**const** char *_name_p_)

Search the global list of network interfaces for an interface with given name and return it.

**Return** Found network interface or NULL if it was not found.

**Parameters**

- `name_p`: Name of the network interface to find.

int **network_interface_set_ip_info**(**struct** *network_interface_t* *_netif_p_, **const** **struct** *inet_if_ip_info_t* *_info_p_)

Set the IP information of given network interface.

**Return** zero(0) or negative error code.

**Parameters**

- `netif_p`: Network interface to get the IP information of.
- `info_p`: IP information to set.

int **network_interface_get_ip_info**(**struct** *network_interface_t* *_netif_p_, **struct** *inet_if_ip_info_t* *_info_p_)

Get the IP information of given network interface.

**Return** zero(0) or negative error code.

**Parameters**

- `netif_p`: Network interface to get the IP information of.
- `info_p`: Read IP information.

**struct network_interface_t**

### Public Members

**const** char ***name_p**

**struct** *inet_if_ip_info_t* **info**

*network_interface_start_t* **start**

*network_interface_stop_t* **stop**

*network_interface_is_up_t* **is_up**

*network_interface_set_ip_info_t* **set_ip_info**

*network_interface_get_ip_info_t* **get_ip_info**

void ***netif_p**

**struct** *network_interface_t* ***next_p**

## `ping` — Ping

### Debug file system commands

One debug file system command is available, located in the directory `inet/ping/`.

| Command | Description |
|---|---|
| `ping <remote host>` | Ping a remote host by given ip address. |

Example output from the shell:

```
$ inet/ping/ping 192.168.1.100
Successfully pinged '192.168.1.100' in 10 ms.
OK
```

---

Source code: src/inet/ping.h, src/inet/ping.c

Test code: tst/inet/ping/main.c

Test coverage: src/inet/ping.c

---

### Functions

int **ping_module_init**(void)

Initialize the ping module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **ping_host_by_ip_address**(**struct** *inet_ip_addr_t* *address_p, **struct** *time_t* *timeout_p, **struct** *time_t* *round_trip_time_p)

Ping host by given ip address. Send an echo request packet to given host and wait for the echo reply packet. No extra payload data is transmitted, only the ICMP header.

**Return** zero(0) or negative error code.

**Parameters**

- `address_p`: IP address of the host to ping.
- `timeout_p`: Number of seconds to wait for the echo reply packet.
- `round_trip_time_p`: The time it took from sending the echo request packet to receiving the echo reply packet. Only valid if this functions returns zero(0).

## `socket` — Internet communication

Sockets are used to communicate over IP networks. TCP and UDP are the most common transport protocols.

No more than one thread may read from a socket at any given moment. The same applies when writing to a socket. The reader and writer may be different threads, though. The behaviour is undefined if more threads use the same

socket simultaneously. The application will likely crash. Add a semaphore to protect the socket if more threads need access to a socket.

Below is a TCP client example that connects to a server and sends data.

```
uint8_t buf[16];
struct socket_t tcp;
struct inet_addr_t local_addr, remote_addr;

/* Set the local and remote addresses. */
inet_aton("192.168.1.103", &local_addr.ip);
local_addr.port = 6000;
inet_aton("192.168.1.106", &remote_addr.ip);
remote_addr.port = 5000;

/* Initialize the socket and connect to the server. */
socket_open_tcp(&tcp);
socket_bind(&tcp, &local_addr);
socket_connect(&tcp, &remote_addr);

/* Send the data. */
memset(buf, 0, sizeof(buf));
socket_write(&tcp, buf, sizeof(buf));

/* Close the connection. */
socket_close(&tcp);
```

And below is the same scenario for UDP.

```
uint8_t buf[16];
struct socket_t udp;
struct socket_addr_t local_addr, remote_addr;

/* Set the local and remote addresses. */
inet_aton("192.168.1.103", &local_addr.ip);
local_addr.port = 6000;
inet_aton("192.168.1.106", &remote_addr.ip);
remote_addr.port = 5000;

/* Initialize the socket and connect to the server. */
socket_open_udp(&udp);
socket_bind(&udp, &local_addr);
socket_connect(&udp, &remote_addr);

/* Send the data. */
memset(buf, 0, sizeof(buf));
socket_send(&udp, buf, sizeof(buf));

/* Close the connection. */
socket_close(&udp);
```

---

Source code: src/inet/socket.h, src/inet/socket.c

---

### Defines

**SOCKET_DOMAIN_INET**

**SOCKET_TYPE_STREAM**
TCP socket type.

**SOCKET_TYPE_DGRAM**
UDP socket type.

**SOCKET_TYPE_RAW**
RAW socket type.

**SOCKET_PROTO_ICMP**

### Functions

int **socket_module_init**(void)
Initialize the socket module. This function will start the lwIP TCP/IP stack. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **socket_open_tcp**(**struct** *socket_t *self_p*)
Initialize given TCP socket.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket to initialize.

int **socket_open_udp**(**struct** *socket_t *self_p*)
Initialize given UDP socket.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket to initialize.

int **socket_open_raw**(**struct** *socket_t *self_p*)
Initialize given RAW socket.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket to initialize.

int **socket_open**(**struct** *socket_t *self_p*, int *domain*, int *type*, int *protocol*)
Initialize given socket.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket to initialize.
- `domain`: Socket domain.
- `type`: Socket type.
- `protocol`: Socket protocol.

int **socket_close**(**struct** *socket_t *self_p*)

Close given socket. No data transfers are allowed on after the socket has been closed.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket to close.

int **socket_bind**(**struct** *socket_t *self_p*, **const struct** *inet_addr_t *local_addr_p*)

Bind given local address to given socket.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket.
- `local_addr_p`: Local address.

int **socket_listen**(**struct** *socket_t *self_p*, int *backlog*)

Listen for connections from remote clients. Only applicable for TCP sockets.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket to listen on.
- `backlog`: Unused.

int **socket_connect**(**struct** *socket_t *self_p*, **const struct** *inet_addr_t *remote_addr_p*)

Connect to given remote address. Connecting a UDP socket sets the default remote address for outgoing datagrams. For TCP a three-way handshake with the remote peer is initiated.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket.
- `remote_addr_p`: Remote address.

int **socket_connect_by_hostname**(**struct** *socket_t *self_p*, **const** char *hostname_p*, uint16_t *port*)

Connect to the remote device with given hostname.

In computer networking, a hostname (archaically nodename) is a label that is assigned to a device connected to a computer network and that is used to identify the device in various forms of electronic communication, such as the World Wide Web.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Socket.
- `hostname_p`: The hostname of the remote device to connect to.
- `port`: Remote device port to connect to.

int **socket_accept**(**struct** *socket_t *self_p*, **struct** *socket_t *accepted_p*, **struct** *inet_addr_t *remote_addr_p*)

Accept a client connect attempt. Only applicable for TCP sockets that are listening for connections.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: TCP socket.
- `accepted_p`: New client socket of the accepted client.
- `remote_addr_p`: Address of the client.

ssize_t **socket_sendto**(**struct** *socket_t *self_p*, **const** void *buf_p, size_t *size*, int *flags*, **const** **struct** *inet_addr_t *remote_addr_p*)

Write data to given socket. Only used by UDP sockets.

**Return** Number of sent bytes or negative error code.

**Parameters**

- `self_p`: Socket to send data on.
- `buf_p`: Buffer to send.
- `size`: Size of buffer to send.
- `flags`: Unused.
- `remote_addr_p`: Remote address to send the data to.

ssize_t **socket_recvfrom**(**struct** *socket_t *self_p*, void *buf_p, size_t *size*, int *flags*, **struct** *inet_addr_t *remote_addr_p*)

Read data from given socket. Only used by UDP sockets.

**Return** Number of received bytes or negative error code.

**Parameters**

- `self_p`: Socket to receive data on.
- `buf_p`: Buffer to read into.
- `size`: Size of buffer to read.
- `flags`: Unused.
- `remote_addr_p`: Remote address to receive data from.

ssize_t **socket_write**(**struct** *socket_t *self_p*, **const** void *buf_p, size_t *size*)

Write data to given TCP or UDP socket. For UDP sockets, `socket_connect()` must have been called prior to calling this function.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: Socket.
- `buf_p`: Buffer to send.
- `size`: Numer of bytes to send.

ssize_t **socket_read**(**struct** *socket_t* *self_p, void *buf_p, size_t size)
Read data from given socket.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: Socket.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **socket_size**(**struct** *socket_t* *self_p)
Get the number of input bytes currently stored in the socket. May return less bytes than number of bytes stored in the channel.

**Return** Number of input bytes in the socket.

**Parameters**

- `self_p`: Socket.

**struct socket_t**

### Public Members

**struct** *chan_t* **base**

int **type**

ssize_t **left**

**struct socket_t::@75::@77::@79 socket_t::common**

**struct** pbuf ***pbuf_p**

**struct** *inet_addr_t* **remote_addr**

int **closed**

**struct socket_t::@75::@77::@80 socket_t::recvfrom**

**struct** tcp_pcb ***pcb_p**

**struct socket_t::@75::@77::@81 socket_t::accept**

**union socket_t::@75::@77 socket_t::u**

int **state**

void ***args_p**

**struct** *thrd_t* ***thrd_p**

**struct socket_t::@75::@78 socket_t::cb**

**struct socket_t::@75 socket_t::input**

**struct socket_t::@76::@82 socket_t::cb**

**struct socket_t::@76 socket_t::output**

void ***pcb_p**

### ssl — Secure socket layer

SSL/TLS based on mbedTLS. Server side sockets works, but not client side.

**Warning:** This module may lead to a false sense of security, as it is implemented by a TLS/SSL novice, me. Use with care!

Simplified server and client side examples to illustrate how to use the module. All error checking is left out to make the example easier to understand. There are links to the full examples further down in this document.

Server side:

```
/* Create the SSL context. */
ssl_context_init(&context, ssl_protocol_tls_v1_0);
ssl_context_load_cert_chain(&context, &certificate[0], &key[0]);

/* Create the TCP listener socket. */
socket_open_tcp(&listener_sock);
socket_bind(&listener_sock, &addr);
socket_listen(&listener_sock, 5);

/* Accept a client.*/
socket_accept(&listener_sock, &sock, &addr);
ssl_socket_open(&ssl_sock,
                &context,
                &sock,
                SSL_SOCKET_SERVER_SIDE,
                NULL);

/* Communicate with the client. */
ssl_socket_read(&ssl_sock, &buf[0], 6);
ssl_socket_write(&ssl_sock, "Goodbye!", 8);
ssl_socket_close(&ssl_sock);
socket_close(&sock);
```

Client side:

```
/* Create the SSL context. */
ssl_context_init(&context, ssl_protocol_tls_v1_0);
ssl_context_load_verify_location(&context, &certificate[0]);

/* Create the TCP socket and connect to the server. */
socket_open_tcp(&sock);
socket_connect(&sock, &addr);
ssl_socket_open(&ssl_sock,
                &context,
                &sock,
                0,
                "foobar.org");
```

(continues on next page)

(continued from previous page)

```
/* Communicate with the client. */
ssl_socket_write(&ssl_sock, "Hello!", 6);
ssl_socket_read(&ssl_sock, &buf[0], 8);
ssl_socket_close(&ssl_sock);
socket_close(&ssl_sock);
```

---

Source code: src/inet/ssl.h, src/inet/ssl.c

Test code: tst/inet/ssl/main.c,

Test coverage: src/inet/ssl.c

Example code: examples/ssl_client/main.c, examples/ssl_server/main.c

---

### Defines

**SSL_SOCKET_SERVER_SIDE**
Server side socket.

### Enums

**enum ssl_protocol_t**
*Values:*

**ssl_protocol_tls_v1_0**

**enum ssl_verify_mode_t**
*Values:*

**ssl_verify_mode_cert_none_t** = 0

**ssl_verify_mode_cert_required_t** = 2

### Functions

int **ssl_module_init**(void)
Initialize the SSL module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **ssl_context_init**(**struct** *ssl_context_t* **self_p*, **enum** *ssl_protocol_t protocol*)
Initialize given SSL context. A SSL context contains settings that lives longer than a socket.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL context to initialize.
- `protocol`: SSL protocol to use.

int **ssl_context_destroy**(**struct** *ssl_context_t* *self_p)

Destroy given SSL context. The context may not be used after it has been destroyed.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL context to destroy.

int **ssl_context_load_cert_chain**(**struct** *ssl_context_t* *self_p, **const** char *cert_p, **const** char *key_p)

Load given certificate chain into given context.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL context.
- `self_p`: Certificate to load.
- `self_p`: Optional key to load. May be NULL.

int **ssl_context_load_verify_location**(**struct** *ssl_context_t* *self_p, **const** char *ca_certs_p)

Load a set of "certification authority" (CA) certificates used to validate other peers' certificates when `verify_mode` is other than `ssl_verify_mode_cert_none_t`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL context.
- `ca_certs_p`: CA certificates.

int **ssl_context_set_verify_mode**(**struct** *ssl_context_t* *self_p, **enum** *ssl_verify_mode_t mode*)

Whether to try to verify other peers' certificates.

Load CA certificates with `ssl_context_load_verify_location()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL context.
- `mode`: Mode to set.

int **ssl_socket_open**(**struct** *ssl_socket_t* *self_p, **struct** *ssl_context_t* *context_p, void *socket_p, int *flags*, **const** char *server_hostname_p)

Initialize given SSL socket with given socket and SSL context. Performs the SSL handshake.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL socket to initialize.
- `context_p`: SSL context to execute in.
- `socket_p`: Socket to wrap in the SSL socket.
- `flags`: Give as `SSL_SOCKET_SERVER_SIDE` for server side sockets. Otherwise 0.

- `server_hostname_p`: The server hostname used by client side sockets to verify the server. Give as NULL to skip the verification. Must be NULL for server side sockets.

int **ssl_socket_close**(**struct** *ssl_socket_t* **self_p*)
Close given SSL socket.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL socket to close.

ssize_t **ssl_socket_write**(**struct** *ssl_socket_t* **self_p*, **const** void **buf_p*, size_t *size*)
Write data to given SSL socket.

**Return** Number of written bytes or negative error code.

**Parameters**

- `self_p`: SSL socket.
- `buf_p`: Buffer to send.
- `size`: Numer of bytes to send.

ssize_t **ssl_socket_read**(**struct** *ssl_socket_t* **self_p*, void **buf_p*, size_t *size*)
Read data from given SSL socket.

**Return** Number of read bytes or negative error code.

**Parameters**

- `self_p`: SSL socket.
- `buf_p`: Buffer to read into.
- `size`: Number of bytes to read.

ssize_t **ssl_socket_size**(**struct** *ssl_socket_t* **self_p*)
Get the number of input bytes currently stored in the SSL socket.

**Return** Number of input bytes in the SSL socket.

**Parameters**

- `self_p`: SSL socket.

**const** char ***ssl_socket_get_server_hostname**(**struct** *ssl_socket_t* **self_p*)
Get the hostname of the server.

**Return** Server hostname or NULL.

**Parameters**

- `self_p`: SSL socket.

int **ssl_socket_get_cipher**(**struct** *ssl_socket_t* **self_p*, **const** char ***cipher_pp*, **const** char ***protocol_pp*, int **number_of_secret_bits_p*)
Get the cipher information.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SSL socket.
- `cipher_pp`: Connection cipher.
- `protocol_pp`: Connection protocol.
- `number_of_secret_bits_p`: Number of secret bits.

**struct ssl_context_t**

#### Public Members

*ssl_protocol_t* **protocol**

void ***conf_p**

int **server_side**

int **verify_mode**

**struct ssl_socket_t**

#### Public Members

**struct** *chan_t* **base**

void ***ssl_p**

void ***socket_p**

### tftp_server — TFTP server

TFTP is a simple file transfer protocol.

Only binary mode is supported.

---

Source code: src/inet/tftp_server.h, src/inet/tftp_server.c

Test code: tst/inet/tftp_server/main.c

Test coverage: src/inet/tftp_server.c

Example code: examples/tftp_server/main.c

---

### Functions

int **tftp_server_init**(**struct** *tftp_server_t* **self_p*, **struct** *inet_addr_t* **addr_p*, int *timeout_ms*, **const** char **name_p*, **const** char **root_p*, void **stack_p*, size_t *stack_size*)

Initialize given TFTP server.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: TFTP server to initialize.
- `addr_p`: Ip address and port of the server.
- `timeout_ms`: Packet reception timeout.
- `name_p`: Name of the server thread.
- `root_p`: File system root path.
- `stack_p`: Server thread stack.
- `stack_size`: Server thread stack size.

int **tftp_server_start**(**struct** *tftp_server_t* **self_p*)
Start given TFTP server.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: TFTP server to start.

**struct tftp_server_t**

#### Public Members

**struct** *inet_addr_t* **addr**

**struct** *socket_t* **listener**

int **timeout_ms**

**const** char ***name_p**

**const** char ***root_p**

void ***stack_p**

size_t **stack_size**

**struct** *thrd_t* ***thrd_p**

### 1.6.6 oam

Operations and maintenence of an application is essential to configure, debug and monitor its operation.

The oam package on Github.

#### `console` — System console

The system console is the default communication channel to an application. The console input and output channels are often terminated by a shell to enable the user to control and debug the application.

Configure the console by changing the *configuration variables* called `CONFIG_START_CONSOLE*`.

---

Source code: src/oam/console.h, src/oam/console.c

Test coverage: src/oam/console.c

### Functions

int **console_module_init**(void)
Initialize the console module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **console_init**(void)
Initialize the console.

**Return** zero(0) or negative error code.

int **console_start**(void)
Start the console.

**Return** zero(0) or negative error code.

int **console_stop**(void)
Stop the console.

**Return** zero(0) or negative error code.

int **console_set_input_channel**(void *chan_p)
Set the pointer to the input channel.

**Return** zero(0) or negative error code.

void ***console_get_input_channel**(void)
Get the pointer to the input channel.

**Return** Input channel or NULL.

void ***console_set_output_channel**(void *chan_p)
Set the pointer to the output channel.

**Return** zero(0) or negative error code.

void ***console_get_output_channel**(void)
Get the pointer to the output channel.

**Return** Output channel or NULL.

### nvm — Non-volatile memory

A non-volatile memory is typically used for long-term persistent storage.

This module implements a singleton non-volatile memory, often on top of an EEPROM or software emulated EEPROM.

---

Source code: src/oam/nvm.h, src/oam/nvm.c

Test coverage: src/oam/nvm.c

---

## Functions

int **nvm_module_init**(void)
Initialize the NVM module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **nvm_mount**(void)
Mount the non-volatile memory.

**Return** zero(0) if the memory was successfully mounted, otherwise negative error code.

int **nvm_format**(void)
Format the non-volatile memory, writing 0xff/erasing to the whole memory. A formatted NVM can always be mounted with `nvm_mount()`.

**Return** zero(0) or negative error code.

ssize_t **nvm_read**(void *_dst_p_, uint32_t _src_, size_t _size_)
Read into given buffer from given NVM address.

**Return** Number of bytes read or negative error code.

**Parameters**

- `dst_p`: Buffer to read into.
- `src`: Address in NVM to read from. Addressing starts at zero(0).
- `size`: Number of bytes to read.

ssize_t **nvm_write**(uint32_t _dst_, **const** void *_src_p_, size_t _size_)
Write given buffer to given NVM address.

**Return** Number of bytes written or negative error code.

**Parameters**

- `dst`: Address in NVM to write to. Addressing starts at zero(0).
- `src_p`: Buffer to write.
- `size`: Number of bytes to write.

ssize_t **nvm_vwrite**(**struct** *iov_uintptr_t* *dst_p, **struct** *iov_t* *src_p, size_t *length*)
Write given buffers to given NVM addresses.

**Return** Number of bytes written or negative error code.

**Parameters**

- `dst_p`: Address ranges in NVM to write to. Addressing starts at zero(0).
- `src_p`: Buffers to write in the same order as in `dst_p`. The size fields are not used.
- `length`: Number of elements in `dst_p` and `src_p`.

### `service` — Services

A service is as a background task. A service is either running or stopped.

### Debug file system commands

Three debug file system commands is available, all located in the directory `oam/service/`.

| Command | Description |
|---|---|
| `list` | List all registered services. |
| `start <service>` | Start given service. |
| `stop <service>` | Stop given service. |

Example output from the shell:

```
$ oam/service/list
NAME                    STATUS
http_server             running
ftp_server              stopped
network_manager         running
OK
$ oam/service/start ftp_server
OK
$ oam/service/stop http_server
OK
$ oam/service/list
NAME                    STATE
http_server             stopped
ftp_server              running
network_manager         running
OK
```

---

Source code: src/oam/service.h, src/oam/service.c

Test code: tst/oam/service/main.c

Test coverage: src/oam/service.c

---

### Defines

**SERVICE_CONTROL_EVENT_START**
Service start event.

**SERVICE_CONTROL_EVENT_STOP**
Serviece stop event.

### Typedefs

**typedef enum** *service_status_t* (***service_get_status_cb_t**)(**struct** *service_t* *self_p)

### Enums

**enum service_status_t**
*Values:*

**service_status_running_t** = 0

**service_status_stopped_t** = 1

### Functions

int **service_module_init**(void)
Initialize the service module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **service_init**(**struct** *service_t* **self_p*, **const** char **name_p*, *service_get_status_cb_t status_cb*)
Initialize a service with given name and status callback.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Service to initialize.
- `name_p`: Name of the service.
- `status_callback`: Callback function returning the service status.

int **service_start**(**struct** *service_t* **self_p*)
Start given service.

The event `SERVICE_CONTROL_EVENT_START` will be written to the control channel of given service and it's up to the service to act on this event. All services should act on all control events.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Service to start.

int **service_stop**(**struct** *service_t* *self_p)
Stop given service.

The event SERVICE_CONTROL_EVENT_STOP will be written to the control channel of given service and it's up to the service to act on this event. All services should act on all control events.

**Return** zero(0) or negative error code.

**Parameters**

- self_p: Service to stop.

int **service_register**(**struct** *service_t* *service_p)
Register given service to the global list of services.

**Return** zero(0) or negative error code.

**Parameters**

- service_p: Service to register.

int **service_deregister**(**struct** *service_t* *service_p)
Deregister given service from the global list of services.

**Return** zero(0) or negative error code.

**Parameters**

- service_p: Service to deregister.

**struct service_t**
*#include* <*service.h*> A service with name and control event channel.

### Public Members

**const** char ***name_p**

**struct** *event_t* **control**

*service_get_status_cb_t* **status_cb**

**struct** *service_t* ***next_p**

## settings — Persistent application settings

Settings are stored in a non-volatile memory (NVM). In other words, settings are perserved after a board reset or power cycle.

Application settings are defined in an ini-file that is used to generate the c source code. A setting has a type, a size, an address and a default value, all defined in the ini-file.

Supported types are:

- int32_t A 32 bits signed integer.
- string An ASCII string.
- blob A chunk of data.

The size is the number of bytes of the value. For the standard integer types the size must be the value returned by *sizeof()*. For strings it is the length of the string, including null termination.

The address for each setting is defined by the user, starting at address 0 and increasing from there.

The build system variable `SETTINGS_INI` contains the path to the ini-file used by the build system.

### Debug file system commands

Four debug file system commands are available, all located in the directory `oam/settings/`.

| Command | Description |
|---|---|
| `list` | Print a list of the current settings. |
| `reset [<name>]` | Reset one or all settings to their default values (the values defined in the ini-file values). |
| `read <name>` | Read the value of setting `<name>`. |
| `write <name> <value>` | Write `<value>` to setting `<name>`. |

Example output from the shell:

```
$ oam/settings/list
NAME                  TYPE      SIZE  VALUE
version               int32_t      4  1
value_1               int32_t      4  24567
value_2               blob_t       4  cafebabe
value_3               string_t    16  foobar
OK
$ oam/settings/read value_1
24567
OK
$ oam/settings/write value_1 -5
OK
$ oam/settings/read value_1
-5
OK
$ oam/settings/reset value_1
OK
$ oam/settings/read value_1
24567
OK
$ oam/settings/reset
OK
$ oam/settings/list
NAME                  TYPE      SIZE  VALUE
version               int32_t      4  1
value_1               int32_t      4  24567
value_2               blob_t       4  cafebabe
value_3               string_t    16  foobar
OK
```

### Example

In this example the ini-file has one setting defined, `foo`. The type is `int32_t`, the address is `0x00`, the size is `4` and the default value is `-4`.

```
[values]
foo = -4

[types]
foo = int32_t

[addresses]
foo = 0x00

[sizes]
foo = 4
```

The settings can be read and written with the functions *settings_read()* and *settings_write()*. Give the generated defines `SETTING_FOO_ADDR` and `SETTING_FOO_SIZE` as arguments to those functions.

```
int my_read_write_foo()
{
    int32_t foo;

    /* Read the foo setting. */
    if (settings_read(&foo,
                      SETTING_FOO_ADDR,
                      SETTING_FOO_SIZE) != 0) {
        return (-1);
    }

    foo -= 1;

    /* Write the foo setting. */
    if (settings_write(SETTING_FOO_ADDR,
                       &foo,
                       SETTING_FOO_SIZE) != 0) {
        return (-1);
    }

    return (0);
}
```

---

Source code: src/oam/settings.h, src/oam/settings.c

Test code: tst/oam/settings/main.c

Test coverage: src/oam/settings.c

---

## Defines

**SETTINGS_AREA_CRC_OFFSET**

## Enums

**enum setting_type_t**

Settings types. Each setting must have be one of these types.

*Values:*

**setting_type_int32_t** = 0

**setting_type_string_t**

**setting_type_blob_t**

### Functions

int **settings_module_init**(void)
Initialize the settings module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

ssize_t **settings_read**(void *_dst_p_, size_t _src_, size_t _size_)
Read the value of given setting by address.

**Return** Number of words read or negative error code.

**Parameters**

- `dst_p`: The read value.
- `src`: Setting source address.
- `size`: Number of words to read.

ssize_t **settings_write**(size_t _dst_, **const** void *_src_p_, size_t _size_)
Write given value to given setting by address.

**Return** Number of words written or negative error code.

**Parameters**

- `dst`: Destination setting address.
- `src_p`: Value to write.
- `size`: Number of bytes to write.

int **settings_reset**(size_t _address_, size_t _size_)
Reset given setting to its default value.

**Return** zero(0) or negative error code.

**Parameters**

- `address`: Setting address.
- `size`: Number of bytes to reset.

ssize_t **settings_read_by_name**(**const** char *_name_p_, void *_dst_p_, size_t _size_)
Read the value of given setting by name.

**Return** Number of words read or negative error code.

**Parameters**

- `name_p`: Setting name.
- `dst_p`: The read value.
- `size`: Size of the destination buffer.

ssize_t **settings_write_by_name**(**const** char *_name_p_, **const** void *_src_p_, size_t _size_)
Write given value to given setting by name.

**Return** Number of words read or negative error code.

**Parameters**

- `name_p`: Setting name.
- `src_p`: Value to write.
- `size`: Number of bytes to write.

int **settings_reset_by_name**(**const** char *_name_p_, size_t _size_)
Reset given setting to its default value.

**Return** zero(0) or negative error code.

**Parameters**

- `name_p`: Setting name.
- `size`: Number of bytes to reset.

int **settings_reset_all**(void)
Reset all settings to their default values.

**Return** zero(0) or negative error code.

**struct setting_t**

### Public Members

**FAR const char* setting_t::name_p**

*setting_type_t* **type**

uint32_t **address**

size_t **size**

## `shell` — Debug shell

The shell is a command line interface where the user can execute various commands to control, debug and monitor its application.

```
> make -s console
$ kernel/sys/info
app:    shell-master built 2017-03-05 21:26 CET by erik.
board:  Arduino Due
mcu:    Atmel SAM3X8E Cortex-M3 @ 84MHz, 96k sram, 512k flash
OK
```

(continues on next page)

(continued from previous page)

```
$ kernel/thrd/list
            NAME         STATE  PRIO   CPU   SCHEDULED  MAX-STACK-USAGE  LOGMASK
         monitor     suspended   -80    0%          22        176/   518     0x0f
            idle         ready   127   99%         594        276/   390     0x0f
            main       current     0    0%         305        540/ 88898     0x00
OK
$ kernel/thrd/set_log_mask foo 0
ERROR(-3)
$ <Ctrl-D>
>
```

The shell module has a few configuration variables that can be used to tailor the shell to the application requirements. Most noticeably is the configuration variable `CONFIG_SHELL_MINIMAL`. If set to `0` all the shell functionality is built; including tab completion, cursor movement, line editing and command history. If set to `1` only the minimal functionality is built; only including tab completion and line editing at the end of the line.

See *Configuration* for a list of all configuration variables.

---

Source code: src/oam/shell.h, src/oam/shell.c

Test code: tst/oam/shell/main.c

Test coverage: src/oam/shell.c

Example code: examples/shell/main.c

---

### Functions

int **shell_module_init**(void)

Initialize the shell module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **shell_init**(**struct** *shell_t* *self_p, void *chin_p, void *chout_p, void *arg_p, **const** char *name_p, **const** char *username_p, **const** char *password_p)

Initialize a shell with given parameters.

**Parameters**

- `chin_p`: The shell input channel. The shell waits for commands on this channel.
- `chout_p`: The shell output channel. The shell writes responses on this channel.
- `arg_p`: User supplied argument passed to all commands.
- `name_p`: The shell thread name.
- `username_p`: Shell login username, or NULL if no username is required to use the shell.
- `password_p`: Shell login password. This field is unused if `username_p` is NULL.

void ***shell_main**(void *arg_p)

The shell main function that listens for commands on the input channel and send response on the output channel. All received commands are passed to the debug file system function `fs_call()` for execution.

**Return** Never returns.

**Parameters**

- `arg_p`: Pointer to the shell arguemnt struct `struct shell_t`. See the struct definition for a description of it's content.

**struct shell_history_elem_t**

### Public Members

**struct** *shell_history_elem_t* ***next_p**

**struct** *shell_history_elem_t* ***prev_p**

char **buf**[1]

**struct shell_line_t**

### Public Members

char **buf**[**CONFIG_SHELL_COMMAND_MAX**]

int **length**

int **cursor**

**struct shell_t**

### Public Members

void ***chin_p**

void ***chout_p**

void ***arg_p**

**const** char ***name_p**

**const** char ***username_p**

**const** char ***password_p**

**struct** *shell_line_t* **line**

**struct** *shell_line_t* **prev_line**

int **carriage_return_received**

int **newline_received**

int **authorized**

**struct** *shell_history_elem_t* ***head_p**

**struct** *shell_history_elem_t* ***tail_p**

**struct** *shell_history_elem_t* ***current_p**

**struct** *shell_line_t* **pattern**

**struct** *shell_line_t* **match**

int **line_valid**

**struct** *circular_heap_t* **heap**

uint8_t **buf**[**CONFIG_SHELL_HISTORY_SIZE**]

**struct shell_t::@121::@122 shell_t::heap**

**struct shell_t::@121 shell_t::history**

### soam — Simba OAM

Simba Operation And Maintenence (SOAM) is a framed debug protocol with enumerated format strings and file system commands. This both saves memory and makes serial port communication more reliable.

Two macros are defined; `OSTR()` and `CSTR()`, both required by the SOAM build system. It is considered good practice to always use these macros, even if SOAM is not used.

- The `OSTR()` macro.

  An output format string.

```
/* Log object. */
log_object_print(NULL, LOG_INFO, OSTR("Hello %s!\r\n"), "Erik");

/* File system command output. */
static int cmd_foo_cb(...)
{
    std_fprintf(chout_p, OSTR("Foo %d!\r\n"), 1);

    return (0);
}

/* Regular printf. */
std_printf(OSTR("Hello 0x%x!\r\n"), 0xbabe);
```

- The `CSTR()` macro.

  A file system command string.

```
fs_command_init(&cmd_foo, CSTR("/foo"), cmd_foo_cb, NULL);
```

### Usage

Enable SOAM by adding `SOAM = yes` to the application makefile.

Connect to the board with `soam.py` instead of a serial terminal program. The only required argument is the string database file.

Here is an example usage of the script. *Ctrl-D* is pressed to exit the script.

```
> soam.py --port /dev/arduino --baudrate 115200 \
      build/arduino_due/soam.soamdb
Welcome to the SOAM shell.

Type help or ? to list commands.

$ kernel/sys/info
```

(continues on next page)

(continued from previous page)

```
app:    soam-master built 2017-03-05 21:26 CET by erik.
board:  Arduino Due
mcu:    Atmel SAM3X8E Cortex-M3 @ 84MHz, 96k sram, 512k flash
OK
$ kernel/thrd/list
            NAME         STATE  PRIO   CPU   SCHEDULED  MAX-STACK-USAGE  LOGMASK
            soam       current    30    0%         112        748/  1542     0x0f
         monitor     suspended   -80    0%          22        176/   518     0x0f
            idle         ready   127   99%         594        276/   390     0x0f
            main     suspended     0    0%         305        540/ 88898     0x00
OK
$ kernel/thrd/set_log_mask foo 0
ERROR(-3)
$ <Ctrl-D>

Bye!
>
```

`OK` is printed by the shell if the file system command returned *zero(0)*, otherwise `ERROR(error code)` is printed.

---

Source code: src/oam/soam.h, src/oam/soam.c

Test code: tst/oam/soam/main.c

Test coverage: src/oam/soam.c

Example code: examples/soam/main.c

---

## Functions

int **soam_module_init**(void)

Initialize the soam module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **soam_init**(**struct** *soam_t* *self_p, void *buf_p, size_t *size*, void *chout_p)

Initialize given soam object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Object to initialize.
- `buf_p`: Transmission buffer.
- `size`: Transmission buffer size.
- `chout_p`: Soam packets are written to this channel.

int **soam_input**(**struct** *soam_t* *self_p, uint8_t *buf_p, size_t *size*)

Process given soam packet.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Soam object.
- `buf_p`: Buffer to input.
- `size`: Size to input in bytes.

ssize_t **soam_write_begin**(**struct** *soam_t* *self_p, int *type*)
Start outputting a soam packet of given type.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Soam object.
- `type`: Packet type.

ssize_t **soam_write_chunk**(**struct** *soam_t* *self_p, **const** void *buf_p, size_t *size*)
Add given chunk of data to current packet.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Soam object.
- `buf_p`: Buffer to output.
- `size`: Size to output in bytes.

ssize_t **soam_write_end**(**struct** *soam_t* *self_p)
Finalize current packet and transmit it.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Soam object.

ssize_t **soam_write**(**struct** *soam_t* *self_p, int *type*, **const** void *buf_p, size_t *size*)
Create and transmit a soam packet of given type and data.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Soam object.
- `type`: Packet type.
- `buf_p`: Buffer to output.
- `size`: Size to output in bytes.

void ***soam_get_log_input_channel**(**struct** *soam_t* *self_p)
Get the log input channel. This channel can be set as output channel of the log module with `log_set_default_handler_output_channel()`.

**Return** Log input channel.

**Parameters**

- `self_p`: Soam object.

void *__soam_get_stdout_input_channel__(__struct__ *soam_t *self_p*)

Get the standard output input channel. This channel can be set as standard output channel of the sys module with `sys_set_stdout()`.

**Return** Standard output input channel.

**Parameters**

- `self_p`: Soam object.

__struct soam_t__

### Public Members

int __is_printf__

uint8_t __transaction_id__

uint8_t *__buf_p__

size_t __size__

ssize_t __pos__

__struct__ *mutex_t* __mutex__

void *__chout_p__

uint8_t __packet_index__

__struct soam_t::@123 soam_t::tx__

__struct__ *chan_t* __stdout_chan__

__struct__ *chan_t* __log_chan__

__struct__ *chan_t* __command_chan__

## `upgrade` — Software upgrade

Upgrade/upload an application over the air (OTA) or using a cable. HTTP, TFTP, Kermit and UDS protocols are supported.

The flash memory is partitioned into two partitions; the bootloader partition and the application partition. The software in the bootloader partition can perform a software upgrade of the application partition by using the erase and write commands.

**Warning:** The WiFi connection is often lost during the erase operation on ESP32. Troubleshooting ongoing...

### Debug file system commands

Five debug file system commands are available, all located in the directory `oam/upgrade/`.

| Command | Description |
|---|---|
| `application/enter` | Enter the appliaction. |
| `application/erase` | Erase the appliaction. May not be called from the application about to be erased. |
| `application/is_valid` | Check is there is a valid application in the memory. |
| `kermit/upload` | Upload a upgrade binary file using the Kermit file transfer protocol. |
| `bootloader/enter` | Enter the bootloader. |

Example output from the shell:

```
$ oam/upgrade/application/is_valid
yes
OK
```

### HTTP requests

Five HTTP requests are available. Form the URL by prefixing them with `http://<hostname>/oam/upgrade/`, ie. `http://<hostname>/oam/upgrade/application/is_valid`.

| Request | Type | Description |
|---|---|---|
| `application/enter` | GET | Enter the appliaction. |
| `application/erase` | GET | Erase the appliaction. May not be called from the application about to be erased. |
| `application/is_valid` | GET | Check is there is a valid application in the memory. |
| `upload` | POST | Upload a upgrade binary file using the Kermit file transfer protocol. |
| `bootloader/enter` | GET | Enter the bootloader. |

### TFTP file transfer

Only upload, aka "put", in binary mode is supported.

### Examples

Here are a few examples of how to upgrade the application using the different supported protocols.

### HTTP

Build and upload the bootloader to the board over the serial port.

```
> make -C bootloader -s BOARD=nano32 run
```

Build the test application and use curl to upload it to the Nano32 over HTTP.

```
> make -C application -s BOARD=nano32
> cd application/build/nano32
> curl http://192.168.0.7/oam/upgrade/application/is_valid
no
> curl --header "Content-Type: application/octet-stream" \
       --data-binary @application.ubin \
       http://192.168.0.7/oam/upgrade/upload
> curl http://192.168.0.7/oam/upgrade/application/is_valid
yes
```

Then start it using HTTP.

```
> curl http://192.168.0.7/oam/upgrade/application/enter
Welcome to the test application!
```

### TFTP

Build and upload the bootloader to the board over the serial port.

```
> make -C bootloader -s BOARD=nano32 run
```

Build the test application and use tftp to upload it to the Nano32 over TFTP.

```
> make -C application -s BOARD=nano32
> cd application/build/nano32
> tftp 192.168.0.7
tftp> mode binary
tftp> put application.ubin
5460544 bytes
tftp> q
```

Then start it using the serial port.

```
> kermit
C-Kermit>connect
$ oam/upgrade/application/is_valid
yes
OK
$ oam/upgrade/application/enter
Welcome to the test application!
```

### Kermit

Build and upload the bootloader to the board over the serial port.

```
> make -s -C bootloader BOARD=arduino_due run
```

Build the test application and use Kermit to upload it to the Arduino Due over the serial port.

```
> make -s -C application BOARD=arduino_due
> cd application/build/arduino_due
> kermit
C-Kermit>connect
```

(continues on next page)

(continued from previous page)

```
$ oam/upgrade/application/is_valid
no
OK
$ oam/upgrade/application/erase
OK
$ oam/upgrade/kermit/upload       # Type '\+c' to return to kermit.
C-Kermit> send application.ubin
```

Then start it using the serial port.

```
C-Kermit> connect
$ oam/upgrade/application/is_valid
OK
yes
$ oam/upgrade/application/enter
Welcome to the test application!
```

---

Source code: src/oam/upgrade.h, src/oam/upgrade.c, src/oam/upgrade

Test code: tst/oam/upgrade/main.c, tst/oam/upgrade/kermit/main.c, tst/oam/upgrade/uds/main.c

Test coverage: src/oam/upgrade.c, src/oam/upgrade

Example code: examples/upgrade/bootloader/main.c, examples/upgrade/application/main.c

---

### Functions

int **upgrade_module_init**(void)
Initialize the upgrade module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **upgrade_bootloader_enter**(void)
Enter the bootloader. This function does not return if all preconditions for entering the bootloader are met.

**Return** zero(0) or negative error code.

int **upgrade_bootloader_stay_set**(void)
Stay in the bootloader after next system reboot.

**Return** zero(0) or negative error code.

int **upgrade_bootloader_stay_clear**(void)
Do not stay in the bootloader after next system reboot.

**Return** zero(0) or negative error code.

int **upgrade_bootloader_stay_get**(void)
Check if the bootlaoder is forced to enter its main loop instead of calling any valid application.

**Return** true(1) if the bootloder shall not call the application, otherwise false(0).

int **upgrade_application_enter**(void)
Enter the application. This function does not return if all preconditions for entering the application are met.

**Return** zero(0) or negative error code.

int **upgrade_application_erase**(void)
Erase the application area.

**Return** zero(0) or negative error code.

int **upgrade_application_is_valid**(int *quick*)
Returns true(1) if there is a valid application in the application area.

**Return** true(1) if a valid application exists in the memory region, otherwise false(0).

**Parameters**

- `quick`: Perform a quick validation. The quick validation is port specific, while the non-quick validation always calculates a checksum of the application and compares it to the expected checksum.

int **upgrade_binary_upload_begin**(void)
Begin an upload transaction of a .ubin file.

**Return** zero(0) or negative error code.

int **upgrade_binary_upload**(**const** void **buf_p*, size_t *size*)
Add data to current upload transaction.

**Return** zero(0) or negative error code.

**Parameters**

- `buf_p`: Buffer to write.
- `size`: Size of the buffer.

int **upgrade_binary_upload_end**(void)
End current upload transaction.

**Return** zero(0) or negative error code.

### 1.6.7 debug

The debug package on Github.

#### harness — Test harness

In software testing, a test harness or automated test framework is a collection of software and test data configured to test a program unit by running it under varying conditions and monitoring its behavior and outputs. It has two main parts: the test execution engine and the test script repository.

This module implements the test execution engine.

The test scripts are part of the build system.

### Stubs

Symbols can be stubbed per C-file using the `STUB()` macro and `STUB` make variable. The `STUB` make variable is a list of source files and the symbols to stub within given file.

For example, stub functions `foo_bar()` and `foo_fie()` in `fum.c` by defining stub functions `STUB(foo_bar)()` and `STUB(foo_fie)()`, and set the make variable `STUB` to `fum.c:foo_bar,foo_fie`.

Prototypes for `foo_bar()` and `foo_fie()` in `foo.h`:

```
/**
 * The bar function.
 */
int foo_bar();

/**
 * The fie function.
 */
int foo_fie();
```

`foo_bar()` and `foo_fie()` called in `fum.c`. Both function calls will call the stubbed version on the respective function.

```
int fum_init()
{
   foo_bar();
   foo_fie();

   return (0);
}
```

The stubbed implementations, often defined in the test suite file `main.c`:

```
int STUB(foo_bar)()
{
    return (0);
}

int STUB(foo_fie)()
{
    return (0);
}
```

And last, add the stubbed symbol to the test suite makefile `Makefile`:

```
STUB = fum.c:foo_bar,foo_fie
```

### Mocking

The test harness also supports function mocking with `mock_write_*()` and `STUB(*)` functions. These functions are generated by the script stub.py.

For example, generate a mock stub of `foo.h` with the command `stub.py generate foo.h ..` Two files are created; `foo_mock.h` and `foo_mock.c`. Include `foo_mock.h` in the test suite and call `mock_write_*()` to prepare the stub for function calls. The stub will verify that all input arguments matches their expected values, and write data to output arguments.

**Note:** The user may have to manually modify parts of the generated stubs to match her use case, as the stub script does not handle all situations properly.

```
#include "simba.h"
#include "foo_mock.h"

static int test_init(void)
{
    /* Make foo_bar() return 1 and foo_fie() 5 once called. */
    mock_write_foo_bar(1);
    mock_write_foo_fie(5);

    BTASSERT(fum_init() == 0);

    return (0);
}

int main()
{
    struct harness_testcase_t testcases[] = {
        { test_init, "test_init" },
        { NULL, NULL }
    };

    sys_start();

    harness_run(testcases);

    return (0);
}
```

Add the stub source file to the list of files to build.

```
STUB = fum.c:foo_bar,foo_fie
SRC += foo_mock.c
```

### Example test suite

Below is an example of a test suite using the harness. It has three test cases; `test_passed`, `test_failed` and `test_skipped`.

The test macro `BTASSERT(condition)` should be used to validate conditions.

```
#include "simba.h"

static int test_passed(void)
{
    /* Return zero(0) when a test case passes. */
    return (0);
}

static int test_failed(void)
{
    /* Return a negative integer when a test case fails. BTASSERT
```

(continues on next page)

(continued from previous page)

```
       will return -1 when the condition is false. */
    BTASSERT(0);

    return (0);
}

static int test_skipped(void)
{
    /* Return a positive integer when a test case is skipped. */
    return (1);
}

int main()
{
    /* Test harness and NULL terminated list of test cases.*/
    struct harness_testcase_t testcases[] = {
        { test_passed, "test_passed" },
        { test_failed, "test_failed" },
        { test_skipped, "test_skipped" },
        { NULL, NULL }
    };

    sys_start();

    harness_run(testcases);

    return (0);
}
```

The output from the test suite is:

```
app:    test_suite-7.0.0 built 2016-07-25 17:38 CEST by erik.
board:  Linux
mcu:    Linux

enter: test_passed
exit: test_passed: PASSED

enter: test_failed
exit: test_failed: FAILED

enter: test_skipped
exit: test_skipped: SKIPPED

            NAME        STATE  PRIO   CPU  LOGMASK
            main      current     0    0%     0x0f
                        ready   127    0%     0x0f
harness report: total(3), passed(1), failed(1), skipped(1)
```

There are plenty of test suites in the tst folder on Github.

---

Source code: src/debug/harness.h, src/debug/harness.c

---

### Defines

**_ASSERTFMT**(fmt, ...)

**_ASSERTHEX**(actual_str, actual, expected_str, expected, actual_size, expected_size)

**BTASSERTRM**(cond, cond_str, res, msg)
Assert given condition. Mark testcase as failed, print an error message and return given value `res` on error.

**BTASSERTR**(cond, cond_str, res, ...)
Assert given condition. Mark testcase as failed, print an error message and return given value `res` on error.

**BTASSERTN**(cond, ...)
Assert given condition. Mark testcase as failed, print an error message and return given value on error.

**BTASSERT**(cond, ...)
Assert given condition. Mark testcase as failed, print an error message and return -1 on error.

**BTASSERTI**(actual, operator, expected)
Compare two integers `actual` and `expected` with given operator `operator`. Mark testcase as failed, print an error message if the condition is not true and return -1 on error.

**BTASSERTM**(actual, expected, size)
Comapre two memory positions `actual` and `expected`. Mark testcase as failed, print an error message if they are not equal and return -1 on error.

**BTASSERTV**(cond, ...)
Assert given condition in a testcase. Mark testcase as failed, print an error message and return on error.

**BTASSERT_IN_RANGE**(value, low, high)
Assert that given value is in given range. Mark testcase as failed, print an error message and return.

**STUB**(function)
Stub given function. Used with the make variable STUB to preprocess object file(s).

### Typedefs

**typedef** int (***harness_testcase_cb_t**)(void)
The testcase function callback.

**Return** zero(0) if the testcase passed, a negative error code if the testcase failed, and a positive value if the testcase was skipped.

**typedef** int (***harness_mock_cb_t**)(void *arg_p, void *buf_p, size_t *size_p)
The read/assert callback function. Modifies a copy of the original mock entry. The modified mock entry is read by read/assert functions.

**Return** true(1) if the mock entry shall be removed, otherwise false(0).

**Parameters**

- `arg_p`: `arg_size` bytes copied from `arg_p` given to `harness_mock_cwrite()`.
- `buf_p`: Mock entry data buffer, equivalent to `buf_p` given to `harness_mock_cwrite()`.
- `size_p`: Mock entry size, equivalent to `size` given to `harness_mock_cwrite()`.

### Functions

int **harness_run**(**struct** *harness_testcase_t *testcases_p*)

Run given testcases in the test harness.

**Return** Never returns.

**Parameters**

- `testcases_p`: An array of testcases to run. The last element in the array must have `callback` and `name_p` set to NULL.

int **harness_expect**(void *chan_p, **const** char *pattern_p, **const struct** *time_t* *timeout_p)

Continiously read from given channel and return when given pattern has been read, or when given timeout occurs.

**Return** Number of bytes read from the channel when match occured, or negative error code.

**Parameters**

- `chan_p`: Channel to read from.
- `pattern_p`: Pattern to wait for.
- `timeout_p`: Timeout, or NULL to wait the default timeout of one second.

ssize_t **harness_mock_write**(**const** char *id_p, **const** void *buf_p, size_t *size*)

Write given data buffer to a mock entry with given id. The mock entry can later be read with `harness_mock_read()` or `harness_mock_try_read()`.

**Return** Number of written words or negative error code.

**Parameters**

- `id_p`: Mock id string to write.

```
NOTE: Only a reference to this string is stored in
      the mock entry.
```

- `buf_p`: Data for given mock id, or NULL if no data shall be written.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.

ssize_t **harness_mock_mwrite**(**const** char *id_p, **const** void *buf_p, size_t *size*, int *length*)

Write given data buffer to a mock entry with given id. The mock entry can later be read `length` times with `harness_mock_read()`, `harness_mock_try_read()` or `harness_mock_assert()`.

**Return** Number of written words or negative error code.

**Parameters**

- `id_p`: Mock id string to write.

```
NOTE: Only a reference to this string is stored in
      the mock entry.
```

- `buf_p`: Data for given mock id, or NULL if no data shall be written.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.
- `length`: Number of times this mock entry will be read/asserted.

ssize_t **harness_mock_cwrite**(**const** char *_id_p_, **const** void *_buf_p_, size_t _size_, _harness_mock_cb_t cb_, void *_arg_p_, size_t _arg_size_)

Write given data buffer to a mock entry with given id. The mock entry can later be read with `harness_mock_read()`, `harness_mock_try_read()` or `harness_mock_assert()` until the callback `cb` returns true(1).

**Return** Number of written words or negative error code.

**Parameters**

- `id_p`: Mock id string to write.

```
NOTE: Only a reference to this string is stored in
      the mock entry.
```

- `buf_p`: Data for given mock id, or NULL if no data shall be written.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.
- `cb`: Callback function called on each read/assert of this mock entry. The mock entry will be removed once the callback returns true(1).
- `arg_p`: Callback argument pointer.
- `arg_size`: Callback argument size.

ssize_t **harness_mock_read**(**const** char *_id_p_, void *_buf_p_, size_t _size_)

Read data from mock entry with given id, and make the testcase fail if the mock id is not found or if given size does not match the size in the mock entry.

**Return** Number of read words or negative error code.

**Parameters**

- `id_p`: Mock id string to read.
- `buf_p`: Buffer to read into, or NULL if no data shall be read.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.

ssize_t **harness_mock_try_read**(**const** char *_id_p_, void *_buf_p_, size_t _size_)

Try to read data from mock entry with given id. The testcase does not fail if the mock entry is missing. However, the test case fails if the mock id is found and the data size does not match.

**Return** Number of read words, -ENOENT if no mock entry was found for given id, or negative error code.

**Parameters**

- `id_p`: Mock id string to read.
- `buf_p`: Buffer to read into, or NULL if no data shall be loaded.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.

int **harness_mock_assert**(**const** char *_id_p_, **const** void *_buf_p_, size_t _size_)

Find mock entry with given id and compare its data to given buffer. The testcase fails if the mock id is not found or on data mismatch.

**Return** zero(0) or negative error code.

**Parameters**

- `id_p`: Mock id string to assert.
- `buf_p`: Buffer with expected data, or NULL if no data shall be compared.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.

ssize_t **harness_mock_write_notify**(**const** char *id_p, **const** void *buf_p, size_t size)

Write given data buffer to a mock entry with given id and notify all raeders that data is available. The `harness_mock_write_notify()` and `harness_mock_read_wait()` functions are useful to mock communication interfaces between threads.

**Return** Number of written words or negative error code.

**Parameters**

- `id_p`: Mock id string to write.

```
NOTE: Only a reference to this string is stored in
      the mock entry.
```

- `buf_p`: Data for given mock id, or NULL if no data shall be written.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.

ssize_t **harness_mock_read_wait**(**const** char *id_p, void *buf_p, size_t size, **struct** *time_t* *time-out_p)

Read data from mock entry with given id. Suspends the current thread if the mock id is not found.

**Return** Number of read words or negative error code.

**Parameters**

- `id_p`: Mock id string to read.
- `buf_p`: Buffer to read into, or NULL if no data shall be read.
- `size`: Buffer size in words, or zero(0) if buf_p is NULL.
- `timeout_p`: Read timeout.

int **harness_set_testcase_result**(int *result*)

Set currently executing testcase result to passed(0), skipped(1) or failed(-1).

**Return** zero(0) or negative error code.

**Parameters**

- `result`: Testcase result to set.

int **harness_get_testcase_result**(void)

Get currently executing testcase result.

**Return** passed(0), skipped(1) or failed(-1).

**struct harness_testcase_t**

#### Public Members

*harness_testcase_cb_t* **callback**

**const** char ***name_p**

### `log` — Logging

The logging module consists of log objects and log handlers. A log object filters log entries and a log handler writes log entries to an output channel.

A log object called "log" and a log handler writing to standard output are created during the log module initialization. The log handler can be replaced by calling *log_set_default_handler_output_channel()*.

Normally one log object is created for each subsystem in an application. This gives the user the power to control which parts of the system to debug and/or monitor at runtime.

It's also possible to print log entries without using log objects, but instead use the current threads' log mask to filter log entries. Just give `NULL` as the first arguement to `log_object_print()`, and the threads' log mask will be used. See *thrd — Threads* details on how to change the threads' log mask.

Sometimes it's useful to write log entries to multiple channels. This is possible by creating and adding another log handler to the log module.

#### Log levels

There are five log levels defined; fatal, error, warning, info and debug. The log levels are defined as `LOG_<upper case level>` in the log module header file.

#### Log entry format

A log entry consists of a timestamp, log level, thread name, log object name and the message. The timestamp is the log entry creation time and the log level is one of fatal, error, warning, info and debug. The thread name is the name of the thread that created the log entry and the log object name is the name of the log object the entry was printed on. The message is a user defined string.

```
<timestamp>:<log level>:<thread name>:<log object name>: <message>
```

#### Debug file system commands

Three debug file system commands are available, all located in the directory `debug/log/`.

| Command | Description |
|---|---|
| `list` | Print a list of all log objects. |
| `print <string>` | Print a log entry using the default log object and log level `LOG_INFO`. This command has no use except to test that the log module works. |
| `set_log_mask <object> <mask>` | Set the log mask to `<mask>` for log object `<object>`. |

Example output from the shell:

```
$ debug/log/list
     OBJECT NAME  MASK
          default  0x0f
OK
$ debug/log/print "Hello World!"
OK
$ debug/log/set_log_mask default 0x1f
OK
$ debug/log/list
     OBJECT NAME  MASK
          default  0x1f
OK
$ debug/log/print "Hello World!!!"
56:info:main:default: Hello World!!!
OK
```

### Example

Here is an example of how to create two log objects; *foo* and *bar*, and then use them and the default log object *default*.

The source code:

```
/* Initialize the log objects foo and bar. */
struct log_object_t foo;
struct log_object_t bar;

log_object_init(&foo, "foo", LOG_UPTO(INFO));
log_object_init(&bar, "bar", LOG_UPTO(DEBUG));

/* Print four log entries. */
log_object_print(&foo, LOG_INFO, OSTR("A foo info message."));
log_object_print(&bar, LOG_INFO, OSTR("A bar info message."));
log_object_print(&bar, LOG_DEBUG, OSTR("A bar debug message."));
log_object_print(NULL, LOG_ERROR, OSTR("A default error message."));
```

All logs are printed from the main thread as can be seen in the third field in the entries in the output below.

```
23.0:info:main:foo: A foo info message.
24.0:info:main:bar: A bar info message.
37.0:debug:main:bar: A bar debug message.
56.0:error:main:default: A default error message.
```

---

Source code: src/debug/log.h, src/debug/log.c

Test code: tst/debug/log/main.c

Test coverage: src/debug/log.c

---

### Defines

**LOG_FATAL**
: An unhandleable error that results in a program crash.

**LOG_ERROR**
A handable error conditions.

**LOG_WARNING**
A warning.

**LOG_INFO**
Generic (useful) information about system operation.

**LOG_DEBUG**
Developer debugging messages.

**LOG_MASK**(level)
Craete a log mask with given level set.

**LOG_UPTO**(level)
Set all levels up to and including given level.

**LOG_ALL**
Set all levels.

**LOG_NONE**
Clear all levels.

### Functions

int **log_module_init**(void)
Initialize the logging module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

int **log_object_init**(**struct** *log_object_t* *self_p, **const** char *name_p, char *mask*)
Initialize given log object with given name and mask.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Log object to initialize.
- `name_p`: Log object name.
- `mask`: Log object mask.

int **log_object_set_log_mask**(**struct** *log_object_t* *self_p, char *mask*)
Set given log mask for given log object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Log object.
- `mask`: Log object mask.

char **log_object_get_log_mask**(**struct** *log_object_t* *self_p)
Get the log mask of given log object.

**Return** Log mask.

**Parameters**

- `self_p`: Log object.

int **log_object_is_enabled_for**(**struct** *log_object_t* **self_p*, int *level*)
Check if given log level is enabled in given log object.

**Return** true(1) if given log level is enabled, false(0) if given log level is disabled, otherwise negative error code.

**Parameters**

- `self_p`: Log object, or NULL to check the level in the thread log mask.
- `level`: Log level to check.

int **log_object_print**(**struct** *log_object_t* **self_p*, int *level*, **const** char **fmt_p*, ...)
Check if given log level is set in the log object mask. If so, format a log entry and write it to all log handlers.

`self_p` may be NULL, and in that case the current thread's log mask is used instead of the log object mask.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Log object, or NULL to use the thread's log mask.
- `level`: Log level.
- `fmt_p`: Log format string.
- `...`: Variable argument list.

int **log_handler_init**(**struct** *log_handler_t* **self_p*, void **chout_p*)
Initialize given log handler with given output channel.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Log handler to initialize.
- `chout_p`: Output handler.

int **log_add_handler**(**struct** *log_handler_t* **handler_p*)
Add given log handler to the list of log handlers. Log entries will be written to all log handlers in the list.

**Return** zero(0) or negative error code.

**Parameters**

- `handler_p`: Log handler to add.

int **log_remove_handler**(**struct** *log_handler_t* **handler_p*)
Remove given log handler from the list of log handlers.

**Return** zero(0) or negative error code.

**Parameters**

- `handler_p`: Log handler to remove.

int **log_add_object**(**struct** *log_object_t* *object_p)
Add given log object to the list of log objects. There are file system commands to list all log objects in the list and also modify their log mask.

**Return** zero(0) or negative error code.

**Parameters**

- `object_p`: Log object to add.

int **log_remove_object**(**struct** *log_object_t* *object_p)
Remove given log object from the list of log objects.

**Return** zero(0) or negative error code.

**Parameters**

- `object_p`: Object to remove.

int **log_set_default_handler_output_channel**(void *chout_p)
Set the output channel of the default log handler.

**Return** zero(0) or negative error code.

**Parameters**

- `chout_p`: Channel to set as the default output channel. May be NULL if no output should be written.

**struct log_handler_t**

#### Public Members

void ***chout_p**

**struct** *log_handler_t* ***next_p**

**struct log_object_t**

#### Public Members

**const** char ***name_p**

char **mask**

**struct** *log_object_t* ***next_p**

### 1.6.8 collections

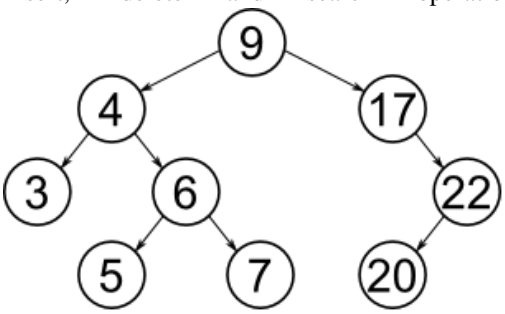
In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently.

The collections package on Github.

### `binary_tree` — Binary tree

A binary search tree consists of nodes, where each node has zero, one or two siblings. The left sibling has a lower value and the right sibling has a higher value than the parent.

Insert, delete and search operations all have the time complexity of O(log n).




---

Source code: src/collections/binary_tree.h, src/collections/binary_tree.c

Test code: tst/collections/binary_tree/main.c

Test coverage: src/collections/binary_tree.c

---

## Functions

int **binary_tree_init**(**struct** *binary_tree_t *self_p*)
Initialize given binary tree.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Binary tree.

int **binary_tree_insert**(**struct** *binary_tree_t *self_p*, **struct** *binary_tree_node_t *node_p*)
Insert given node into given binary tree.

There can not be two or more nodes in the tree with the same key. This function returns -1 if a node with the same key is already in the binary tree.

**Return** zero(0) on success, -1 if a node with the same key is already in the binary tree, otherwise negative error code.

**Parameters**

- `self_p`: Binary tree to insert the node into.
- `node_p`: Node to insert.

int **binary_tree_delete**(**struct** *binary_tree_t *self_p*, int *key*)
Delete given node from given binary tree.

**Return** zero(0) on success, -1 if the node was not found, otherwise negative error code.

**Parameters**

- `self_p`: Binary tree to delete the node from.
- `key`: Key of the node to delete.

**struct** *binary_tree_node_t* ***binary_tree_search**(**struct** *binary_tree_t* **self_p*, int *key*)
Search the binary tree for the node with given key.

**Return** Pointer to found node or NULL if a node with given key was not found in the tree.

**Parameters**

- `self_p`: Binary tree to search in.
- `key`: Key of the binary tree node to search for.

void **binary_tree_print**(**struct** *binary_tree_t* **self_p*)
Print given binary tree.

**Parameters**

- `self_p`: Binary tree to print.

**struct binary_tree_node_t**

#### Public Members

int **key**

int **height**

**struct** *binary_tree_node_t* ***left_p**

**struct** *binary_tree_node_t* ***right_p**

**struct binary_tree_t**

#### Public Members

**struct** *binary_tree_node_t* ***root_p**

### `bits` — Bitwise operations

Source code: src/collections/bits.h

Test code: tst/collections/bits/main.c

---

#### Functions

**static** uint32_t **bits_mask_32**(int *width*)
Create a bit mask of given width.

**Return** The bit mask.

**Parameters**

- `width`: Width in bits.

**static** uint32_t **bits_insert_32** (uint32_t *dst*, int *position*, int *size*, uint32_t *src*)
Insert given number of bits into another value at given position.

For example, `bits_insert_32(0xffffffff, 4, 8, 0x12)` would return `0xfffff12f`.

**Return** The resulting value of the insertion.

**Parameters**

- `dst`: Value to insert into.
- `position`: Bit position, counted from LSB, in `dst` where to insert `src`, 0-31.
- `size`: Number of bits to insert. 0-31.
- `src`: Value to insert into dst.

uint8_t **bits_reverse_8** (uint8_t *value*)
Reverse the order of given 8 bits.

**Return** Reversed bits.

uint16_t **bits_reverse_16** (uint16_t *value*)
Reverse the order of given 16 bits.

**Return** Reversed bits.

uint32_t **bits_reverse_32** (uint32_t *value*)
Reverse the order of given 32 bits.

**Return** Reversed bits.

### `circular_buffer` — Circular buffer

Source code: src/collections/circular_buffer.h, src/collections/circular_buffer.c

Test code: tst/collections/circular_buffer/main.c

Test coverage: src/collections/circular_buffer.c

---

### Functions

int **circular_buffer_init** (**struct** *circular_buffer_t* **self_p*, void **buf_p*, size_t *size*)
Initialize given circular buffer.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Circular buffer to initialize.
- `buf_p`: Memory buffer.
- `size`: Size of the memory buffer.

ssize_t **circular_buffer_write**(**struct** *circular_buffer_t* *self_p, **const** void *buf_p, size_t *size*)
Write data to given circular buffer.

**Return** Number of bytes written or negative error code.

**Parameters**

- `self_p`: Circular buffer.
- `buf_p`: Memory buffer to write.
- `size`: Size of the memory buffer.

ssize_t **circular_buffer_read**(**struct** *circular_buffer_t* *self_p, void *buf_p, size_t *size*)
Read data from given circular buffer.

**Return** Number of bytes read or negative error code. The buffer is empty if zero(0) is returned.

**Parameters**

- `self_p`: Circular buffer to free to.
- `buf_p`: Memory buffer to read into.
- `size`: Size of the memory buffer.

ssize_t **circular_buffer_used_size**(**struct** *circular_buffer_t* *self_p)
Returns the number of used bytes in given circular buffer.

**Return** Number of used bytes.

**Parameters**

- `self_p`: Circular buffer.

ssize_t **circular_buffer_unused_size**(**struct** *circular_buffer_t* *self_p)
Returns the number of unused bytes in given circular buffer.

**Return** Number of unused bytes.

**Parameters**

- `self_p`: Circular buffer.

ssize_t **circular_buffer_skip_front**(**struct** *circular_buffer_t* *self_p, size_t *size*)
Skip given number of written bytes at the front of the buffer.

**Return** Number of skipped bytes or negative error code.

**Parameters**

- `self_p`: Circular buffer.
- `size`: Number of bytes to skip.

ssize_t **circular_buffer_reverse_skip_back**(**struct** *circular_buffer_t* *self_p, size_t *size*)
Skip given number of written bytes at the back of the buffer.

**Return** Number of skipped bytes or negative error code.

**Parameters**

- `self_p`: Circular buffer.
- `size`: Number of bytes to skip.

ssize_t **circular_buffer_array_one**(**struct** *circular_buffer_t* *self_p, void **buf_pp, size_t *size*)
Get a pointer to the next byte to read from the buffer. Use `circular_buffer_array_two()` to get the second array, if there is a wrap around.

**Return** Number of bytes in array or negative error code.

**Parameters**

- `self_p`: Circular buffer.
- `buf_pp`: A poitner to the start of the array. Only valid if the return value is greater than zero(0).
- `size`: Number of bytes asked for.

ssize_t **circular_buffer_array_two**(**struct** *circular_buffer_t* *self_p, void **buf_pp, size_t *size*)
Get a pointer to the next byte to read from the buffer, following a wrap around.

**Return** Number of bytes in array or negative error code.

**Parameters**

- `self_p`: Circular buffer.
- `buf_pp`: A poitner to the start of the array. Only valid if the return value is greater than zero(0).
- `size`: Number of bytes asked for.

**struct circular_buffer_t**

#### Public Members

char ***buf_p**

size_t **size**

size_t **writepos**

size_t **readpos**

### `fifo` — First In First Out queuing

Source code: src/collections/fifo.h

Test code: tst/collections/fifo/main.c

---

### Defines

**FIFO_DEFINE_TEMPLATE**(type)
Define the fifo structure and functions for a given type.

```
FIFO_DEFINE_TEMPLATE(int);

int foo()
{
    struct fifo_int_t fifo;
    int buf[4];
    int value;

    fifo_init_int(&fifo, buf, membersof(buf));

    // Put a value into the fifo.
    value = 10;
    fifo_put_int(&fifo, &value);

    // Get the value from the fifo.
    fifo_get_int(&fifo, &value);

    // Prints 'value = 10'.
    std_printf(FSTR("value=  %d\r\n", value));
}
```

**Parameters**

- `type`: Type of the elements in the defined fifo.

## Functions

**static** int **fifo_init** (**struct** *fifo_t* *self_p*, int *max*)
Initialize given fifo.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Fifo to initialize.
- `max`: Maximum number of elements in the fifo.

**static** int **fifo_put** (**struct** *fifo_t* *self_p*)
Put an element in the fifo.

**Return** Added element index in fifo, or -1 if there are no free positions.

**Parameters**

- `self_p`: Initialized fifo.

**static** int **fifo_get** (**struct** *fifo_t* *self_p*)
Get the next element from the fifo.

**Return** The fetched element index in fifo , or -1 if the fifo is empty.

**Parameters**

- `self_p`: Initialized fifo.

**struct fifo_t**

### Public Members

int **rdpos**

int **wrpos**

void *__buf_p__

int **max**

## hash_map — Hash map

Source code: src/collections/hash_map.h, src/collections/hash_map.c

Test code: tst/collections/hash_map/main.c

Test coverage: src/collections/hash_map.c

---

### Typedefs

**typedef** int (*__hash_map_hash_t__)(longptr_t key)

### Functions

int **hash_map_init**(**struct** *hash_map_t* **self_p*, **struct** *hash_map_bucket_t* **buckets_p*, size_t *buckets_max*, **struct** *hash_map_entry_t* **entries_p*, size_t *entries_max*, *hash_map_hash_t hash*)

Initialize hash map with given parameters.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized hash map.
- `buckets_p`: Array of buckets.
- `buckets_max`: Number of entries in `buckets_p`.
- `entries_p`: Array of empty entries.
- `entries_max`: Number of entries in `entries_p`.
- `hash`: Hash function.

int **hash_map_add**(**struct** *hash_map_t* **self_p*, longptr_t *key*, longptr_t *value*)

Add given key-value pair into hash map. Overwrites old value if the key is already present in map.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized hash map.
- `key`: Key to add.
- `value`: Value to insert for key.

int **hash_map_remove**(**struct** *hash_map_t* *self_p, longptr_t *key*)
Remove given key from hash map.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized hash map.
- `key`: Key to remove.

int **hash_map_get**(**struct** *hash_map_t* *self_p, longptr_t *key*, longptr_t *value_p)
Get value for given key.

**Return** zero(0) if the key was found, otherwise negative error code.

**Parameters**

- `self_p`: Initialized hash map.
- `key`: Key to find.
- `value_p`: Value found for given key. Unmodified if the key was not found.

**struct hash_map_entry_t**

#### Public Members

**struct** *hash_map_entry_t* ***next_p**

longptr_t **key**

longptr_t **value**

**struct hash_map_bucket_t**

#### Public Members

**struct** *hash_map_entry_t* ***list_p**

**struct hash_map_t**

#### Public Members

**struct** *hash_map_bucket_t* ***buckets_p**

size_t **buckets_max**

**struct** *hash_map_entry_t* ***entries_p**

*hash_map_hash_t* **hash**

### `list` — Abstract lists

Source code: src/collections/list.h

### Functions

int **list_init**(**struct** *list_t* *self_p*)
Initialize given singly linked list object.

Elements in the list must have a `struct list_elem_t` as their first struct member.

An element can only be part of one list at a time.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: List object to initialize.

void ***list_peek_head**(**struct** *list_t* *self_p*)
Peek at the first element in the list.

**Return** First element of the list, or NULL if the list is empty.

**Parameters**

- `self_p`: List object.

int **list_add_head**(**struct** *list_t* *self_p*, void *elem_p*)
Add given element to the beginning of given list.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: List object.
- `elem_p`: Element to add.

int **list_add_tail**(**struct** *list_t* *self_p*, void *elem_p*)
Add given element to the end of given list.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: List object.
- `elem_p`: Element to add.

void ***list_remove_head**(**struct** *list_t* *self_p*)
Get the first element of given list and then remove it from given list.

**Return** Removed element, or NULL if the list was empty.

**Parameters**

- `self_p`: List object.

void ***list_remove**(**struct** *list_t* *self_p*, void *elem_p*)
Remove given element from given list.

**Return** Removed element, or NULL if the element was not found.

**Parameters**

- `self_p`: List object.
- `elem_p`: Element to remove.

int **list_iter_init**(**struct** *list_iter_t* *self_p, **struct** *list_t* *list_p)
Initialize given iterator object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Iterator to initialize.
- `list_p`: List object to iterate over.

void ***list_iter_next**(**struct** *list_iter_t* *self_p)
Get the next element from given iterator object.

**Return** Next element, or NULL on end of list.

**Parameters**

- `self_p`: Iterator object.

**struct list_elem_t**
*#include* <*list.h*> Singly linked list elements must have this struct as their first member.

#### Public Members

**struct** *list_elem_t* ***next_p**

**struct list_t**
*#include* <*list.h*> A singly linked list.

#### Public Members

**struct** *list_elem_t* ***head_p**

**struct** *list_elem_t* ***tail_p**

**struct list_iter_t**
*#include* <*list.h*> A singly linked list iterator.

#### Public Members

**struct** *list_elem_t* ***next_p**

### 1.6.9 alloc

Memory management is the act of managing computer memory. The essential requirement of memory management is to provide ways to dynamically allocate portions of memory to programs at their request, and free it for reuse when no longer needed.

The alloc package on Github.

### `circular_heap` — Circular heap

The circular heap is a dynamic memory allocator allocating buffers in a circular buffer. This puts a restriction on the user to free allocated buffers in the same order as they were allocated. This allocator is useful if you know the allocation order and need a low memory overhead on each allocated buffer and no memory fragmentation.

Below is an example of the internal state of a circular heap when buffers are allocated and freed.

1. After initialization *begin*, *alloc* and *free* have the same value. All memory is available for allocation.

```
begin
alloc
 free                                                  end
  |-----------------------------------------------------|
```

2. Allocating a buffer increments *alloc*.

```
begin
 free                      alloc                       end
  |=========================|---------------------------|
```

3. Allocating another buffer increments *alloc* once again.

```
begin
 free                                       alloc       end
  |=========================================|----------|
```

4. Freeing the first buffer increments *free* to the position of the first *alloc*.

```
begin                       free           alloc       end
  |-------------------------|================|----------|
```

5. Allocating a buffer that is bigger than the available space between *alloc* and *end* results in a buffer starting at *begin*. The memory between the old *alloc* and *end* will be unused.

```
begin               alloc   free                        end
  |=================|------|================|oooooooooo|
```

6. Freeing the second buffer increments *free* to the position of the second *alloc*.

```
begin               alloc                     free      end
  |=================|---------------------|oooooooooo|
```

7. Freeing the third buffer sets *free* to *alloc*. All memory is available for allocation once again.

```
                    alloc
begin                free                               end
  |------------------|---------------------------------|
```

8. Done!

---

Source code: src/alloc/circular_heap.h, src/alloc/circular_heap.c

Test code: tst/alloc/circular_heap/main.c

Test coverage: src/alloc/circular_heap.c

---

### Functions

int **circular_heap_init**(**struct** *circular_heap_t* *self_p, void *buf_p, size_t size)
Initialize given circular heap. Buffers must be freed in the same order as they were allocated.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Circular heap to initialize.
- `buf_p`: Memory buffer to use for the circular heap.
- `size`: Size of the memory buffer.

void ***circular_heap_alloc**(**struct** *circular_heap_t* *self_p, size_t size)
Allocate a buffer of given size from given circular heap.

**Return** Pointer to allocated buffer, or NULL if no memory could be allocated.

**Parameters**

- `self_p`: Circular heap to allocate from.
- `size`: Number of bytes to allocate.

int **circular_heap_free**(**struct** *circular_heap_t* *self_p, void *buf_p)
Free the oldest allocated buffer, previously allocated with `circular_heap_alloc()`.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Circular heap to free to.
- `buf_p`: Buffer to free. Must be the oldest allocated buffer.

**struct circular_heap_t**

#### Public Members

void ***begin_p**

void ***end_p**

void ***alloc_p**

void ***free_p**

## heap — Heap

Source code: src/alloc/heap.h, src/alloc/heap.c

Test code: tst/alloc/heap/main.c

Test coverage: src/alloc/heap.c

### Defines

**HEAP_FIXED_SIZES_MAX**
Number of fixed size buffer sizes.

### Functions

int **heap_init**(**struct** *heap_t* *self_p, void *buf_p, size_t *size*, size_t *sizes*[*HEAP_FIXED_SIZES_MAX*])
Initialize given heap.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Heap to initialize.
- `buf_p`: Heap memory buffer.
- `size`: Size of the heap memory buffer.
- `sizes`: Fixed buffer sizes configuration.

void *__heap_alloc__(**struct** *heap_t* *self_p, size_t *size*)
Allocate a buffer of given size from given heap. Tries to allocate a fixed size buffer, and allocates a buffer from the dynamic heap if the requested buffer size is greater than the biggest fixed size buffer.

**Return** Pointer to allocated buffer, or NULL if no memory could be allocated.

**Parameters**

- `self_p`: Heap to allocate from.
- `size`: Number of bytes to allocate.

int **heap_free**(**struct** *heap_t* *self_p, void *buf_p)
Decrement the buffer share counter by one and free the buffer if the count becomes zero(0).

**Return** Share count after the free, or negative error code.

**Parameters**

- `self_p`: Heap of given buffer.
- `buf_p`: Memory buffer to free.

int **heap_share**(**struct** *heap_t* *self_p, **const** void *buf_p, int *count*)
Share given buffer `count` times.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Heap of given buffer.
- `buf_p`: Buffer to share.
- `count`: Share count.

**struct heap_fixed_t**

#### Public Members

void ***free_p**

size_t **size**

**struct heap_dynamic_t**

#### Public Members

void ***free_p**

**struct heap_t**

*#include* <*heap.h*> The heap struct.

#### Public Members

void ***buf_p**

size_t **size**

void ***next_p**

**struct** *heap_fixed_t* **fixed**[**HEAP_FIXED_SIZES_MAX**]

**struct** *heap_dynamic_t* **dynamic**

**struct** *mutex_t* **mutex**

### 1.6.10 text

Text parsing, editing and colorization.

The text package on Github.

#### color — ANSI colors

Source code: src/text/color.h

---

#### Defines

**COLOR_RESET**

**COLOR_BOLD_ON**

**COLOR_ITALICS_ON**

**COLOR_UNDERLINE_ON**

**COLOR_INVERSE_ON**

**COLOR_STRIKETHROUGH_ON**

**COLOR_BOLD_OFF**

**COLOR_ITALICS_OFF**

**COLOR_UNDERLINE_OFF**

**COLOR_INVERSE_OFF**

**COLOR_STRIKETHROUGH_OFF**

**COLOR_FOREGROUND_BLACK**

**COLOR_FOREGROUND_RED**

**COLOR_FOREGROUND_GREEN**

**COLOR_FOREGROUND_YELLOW**

**COLOR_FOREGROUND_BLUE**

**COLOR_FOREGROUND_MAGENTA**

**COLOR_FOREGROUND_CYAN**

**COLOR_FOREGROUND_WHITE**

**COLOR_FOREGROUND_DEFAULT**

**COLOR_BACKGROUND_BLACK**

**COLOR_BACKGROUND_RED**

**COLOR_BACKGROUND_GREEN**

**COLOR_BACKGROUND_YELLOW**

**COLOR_BACKGROUND_BLUE**

**COLOR_BACKGROUND_MAGENTA**

**COLOR_BACKGROUND_CYAN**

**COLOR_BACKGROUND_WHITE**

**COLOR_BACKGROUND_DEFAULT**

**COLOR**(...)

### `configfile` — Configuration file (INI-file)

The INI file format is an informal standard for configuration files for some platforms or software. INI files are simple text files with a basic structure composed of sections, properties, and values.

More information on Wikipedia.

#### File format description

- Line terminators: `\n`, `\r\n` or `\n\r`.
- Opening bracket (`[`) at the beginning of a line indicates a section. The section name is all characters until a closing bracket (`]`).
- A property line starts with its name, then a colon (`:`) or equal sign (=), and then the value.
- Semicolon (`;`) or number sign (#) at the beginning of a line indicate a comment.

### Example file

```
; last modified 1 April 2001 by John Doe
[owner]
name = John Doe
organization = Acme Widgets Inc.

[database]
; use IP address in case network name resolution is not working
server = 192.0.2.62
port = 143
file = "payroll.dat"
```

---

Source code: src/text/configfile.h, src/text/configfile.c

Test code: tst/text/configfile/main.c

Test coverage: src/text/configfile.c

---

### Functions

int **configfile_init**(**struct** *configfile_t* *self_p, char *buf_p, size_t *size*)
Initialize given configuration file object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Object to initialize.
- `buf_p`: Configuration file contents as a NULL terminated string.
- `size`: Size of the configuration file contents.

int **configfile_set**(**struct** *configfile_t* *self_p, **const** char *section_p, **const** char *property_p, **const** char *value_p)
Set the value of given property in given section.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized parser.
- `section_p`: Section to set the property from.
- `property_p`: Property to set the value for.
- `value_p`: NULL terminated value to set.

char ***configfile_get**(**struct** *configfile_t* *self_p, **const** char *section_p, **const** char *property_p, char *value_p, int *length*)
Get the value of given property in given section.

**Return** Value pointer or NULL on failure.

**Parameters**

- `self_p`: Initialized parser.
- `section_p`: Section to get the property from.
- `property_p`: Property to get the value for.
- `value_p`: Value of given property in given section.
- `size`: Size of the value buffer.

int **configfile_get_long**(**struct** *configfile_t* *self_p, **const** char *section_p, **const** char *property_p, long *value_p)

Get the value of given property in given section, converted to an integer.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized parser.
- `section_p`: Section to get the property from.
- `property_p`: Property to get the value for.
- `value_p`: Value of given property in given section.

int **configfile_get_float**(**struct** *configfile_t* *self_p, **const** char *section_p, **const** char *property_p, float *value_p)

Get the value of given property in given section, converted to a float.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Initialized parser.
- `section_p`: Section to get the property from.
- `property_p`: Property to get the value for.
- `value_p`: Value of given property in given section.

**struct configfile_t**

#### Public Members

char ***buf_p**

size_t **size**

### `emacs` — Emacs text editor

Emacs is a text editor originally written by Richard Stallman and Guy L. Steele, Jr. in 1976. This module contains a minimal functional Emacs called Atto.

Help and key bindings: https://github.com/eerimoq/atto#atto-key-bindings

Atto Emacs project on GitHub: https://github.com/hughbarney/atto

Source code: src/text/emacs.h, src/text/emacs.c

---

### Functions

int **emacs**(**const** char *path_p, void *chin_p, void *chout_p)
Start emacs.

**Return** zero(0) or negative error code.

**Parameters**

- path_p: File to open or NULL.
- chin_p: Input channel or NULL for standard input.
- chout_p: Output channel or NULL for standard output.

## re — Regular expressions

Source code: src/text/re.h, src/text/re.c

Test code: tst/text/re/main.c

Test coverage: src/text/re.c

---

### Defines

**RE_IGNORECASE**
Perform case-insensitive matching; expressions like [A-Z] will match lowercase letters, too.

**RE_DOTALL**
Make the '.' special character match any character at all, including a newline; without this flag, '.' will match anything except a newline.

**RE_MULTILINE**
When specified, the pattern character '^' matches at the beginning of the string and at the beginning of each line (immediately following each newline); and the pattern character '$' matches at the end of the string and at the end of each line (immediately preceding each newline). By default, '^' matches only at the beginning of the string, and '$' only at the end of the string and immediately before the newline (if any) at the end of the string.

### Functions

int **re_module_init**(void)
Initialize the re module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

char *__re_compile__(char **compiled_p*, __const__ char **pattern_p*, char *flags*, size_t *size*)
Compile given pattern.

Pattern syntax:

- `'.'` - Any character.
- `'^'` - Beginning of the string (**not yet supported**).
- `'$'` - End of the string (**not yet supported**).
- `'?'` - Zero or one repetitions (greedy).
- `'*'` - Zero or more repetitions (greedy).
- `'+'` - One or more repetitions (greedy).
- `'??'` - Zero or one repetitions (non-greedy).
- `*?` - Zero or more repetitions (non-greedy).
- `+?` - One or more repetitions (non-greedy).
- `{m}` - Exactly `m` repetitions.
- `\\` - Escape character.
- `[]` - Set of characters.
- `'|'` - Alternatives (**not yet supported**).
- `(...)` - Groups (**not yet supported**).
- `\\d` - Decimal digits `[0-9]`.
- `\\w` - Alphanumerical characters `[a-ZA-Z0-9_]`.
- `\\s` - Whitespace characters `[ \t\r\n\f\v]`.

**Return** Compiled patten, or NULL if the compilation failed.

**Parameters**

- `compiled_p`: Compiled regular expression pattern.
- `pattern_p`: Regular expression pattern.
- `flags`: A combination of the flags `RE_IGNORECASE`, `RE_DOTALL` and `RE_MULTILINE` (`RE_MULTILINE` is **not yet supported**).
- `size`: Size of the compiled buffer.

ssize_t __re_match__(__const__ char **compiled_p*, __const__ char **buf_p*, size_t *size*, __struct__ *re_group_t* **groups_p*, size_t **number_of_groups_p*)
Apply given regular expression to the beginning of given string.

**Return** Number of matched bytes or negative error code.

**Parameters**

- `compiled_p`: Compiled regular expression pattern. Compile a pattern with `re_compile()`.
- `buf_p`: Buffer to apply the compiled pattern to.
- `size`: Number of bytes in the buffer.
- `groups_p`: Read groups or NULL.

- `number_of_groups_p`: Number of read groups or NULL.

**struct re_group_t**

#### Public Members

**const** char ***buf_p**

ssize_t **size**

### std — Standard functions

Source code: src/text/std.h, src/text/std.c

Test code: tst/text/std/main.c

Test coverage: src/text/std.c

---

#### Functions

int **std_module_init**(void)

Initialize the std module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

ssize_t **std_sprintf**(char **dst_p*, far_string_t *fmt_p*, ...)

Format and write data to destination buffer. The buffer must be big enough to fit the formatted string. The output is null terminated.

A format specifier has this format:

%[flags][width][length]specifier

where

- flags: `0` or `-`
- width: `0..127`
- length: `l` for long or nothing
- specifier: `c`, `s`, `S`, `d`, `i`, `u`, `x` or `f`

The `S` specifier expects a far string (`far_string_t`) argument. Other specifiers have their usual definition.

**Return** Length of the string written to the destination buffer, not including the null termination, or negative error code.

**Parameters**

- `dst_p`: Destination buffer. The formatted string is written to this buffer.
- `fmt_p`: Format string.

- `...`: Variable arguments list.

ssize_t **std_snprintf**(char **dst_p*, size_t *size*, far_string_t *fmt_p*, ...)
Format and write data to given buffer. The output is null terminated.

**Return** Length of the string written to the destination buffer, not including the null termination, -ENOMEM if the string does not fit in the destination buffer, or other negative error code.

**Parameters**

- `dst_p`: Destination buffer. The formatted string is written to this buffer.
- `size`: Size of the destination buffer.
- `fmt_p`: Format string.
- `...`: Variable arguments list.

ssize_t **std_vsprintf**(char **dst_p*, far_string_t *fmt_p*, va_list **ap_p*)
Format and write data to given buffer. The output is null terminated.

**Return** Length of the string written to the destination buffer, not including the null termination, or negative error code.

**Parameters**

- `dst_p`: Destination buffer. The formatted string is written to this buffer.
- `fmt_p`: Format string.
- `ap_p`: Variable arguments list.

ssize_t **std_vsnprintf**(char **dst_p*, size_t *size*, far_string_t *fmt_p*, va_list **ap_p*)
Format and write data to given buffer. The output is null terminated.

**Return** Length of the string written to the destination buffer, not including the null termination, -ENOMEM if the string does not fit in the destination buffer, or other negative error code.

**Parameters**

- `dst_p`: Destination buffer. The formatted string is written to this buffer.
- `size`: Size of the destination buffer.
- `fmt_p`: Format string.
- `ap_p`: Variable arguments list.

ssize_t **std_printf**(far_string_t *fmt_p*, ...)
Format and print data to standard output. The output is not null terminated.

See `std_sprintf()` for the the format string specification.

**Return** Number of characters written to standard output, or negative error code.

**Parameters**

- `fmt_p`: Format string.
- `...`: Variable arguments list.

ssize_t **std_vprintf**(far_string_t *fmt_p*, va_list **ap_p*)
Format and print data to standard output. The output is not null terminated.

See `std_sprintf()` for the the format string specification.

**Return** Number of characters written to standard output, or negative error code.

**Parameters**

- `fmt_p`: Format string.
- `ap_p`: Variable arguments list.

ssize_t **std_fprintf**(void **chan_p*, far_string_t *fmt_p*, ...)
Format and print data to given channel. The output is not null terminated.

See `std_sprintf()` for the the format string specification.

**Return** Number of characters written to given channel, or negative error code.

**Parameters**

- `chan_p`: Output channel.
- `fmt_p`: Format string.
- `...`: Variable arguments list.

ssize_t **std_vfprintf**(void **chan_p*, far_string_t *fmt_p*, va_list **ap_p*)
Format and print data to given channel. The output is not null terminated.

See `std_sprintf()` for the the format string specification.

**Return** Number of characters written to given channel, or negative error code.

**Parameters**

- `chan_p`: Output channel.
- `fmt_p`: Format string.
- `...`: Variable arguments list.

ssize_t **std_printf_isr**(far_string_t *fmt_p*, ...)
Format and print data to standard output from interrupt context or with the system lock taken. The output is not null terminated.

See `std_sprintf()` for the the format string specification.

**Return** Number of characters written to standard output, or negative error code.

**Parameters**

- `fmt_p`: Format string.
- `...`: Variable arguments list.

ssize_t **std_fprintf_isr**(void **chan_p*, far_string_t *fmt_p*, ...)
Format and print data to given channel from interrupt context or with the system lock taken. The output is not null terminated.

See `std_sprintf()` for the the format string specification.

**Return** Number of characters written to given channel, or negative error code.

**Parameters**

- `chan_p`: Output channel.
- `fmt_p`: Format string.
- `...`: Variable arguments list.

**const** char ***std_strtolb**(**const** char **str_p*, long **value_p*, int *base*)
Convert given string to an integer in given base.

**Return** Pointer to the next byte, or NULL if no value was found.

**Parameters**

- `str_p`: Integer string.
- `value_p`: Parsed integer.
- `base`: The base of the value to parse. One of 16, 10, 8 or 2, or 0 to select base based on the string prefix. Supported string prefixes are "0x" for hexadecimal numbers, "0" for octal numbers and "0b" for binary numbers.

**const** char ***std_strtol**(**const** char **str_p*, long **value_p*)
Convert given string to an integer with base selection based on the string prefix.

**Return** Pointer to the next byte, or NULL if no value was found.

**Parameters**

- `str_p`: Integer string.
- `value_p`: Parsed integer.

**const** char ***std_strtod**(**const** char **str_p*, double **value_p*)
Convert given string to a double.

**Return** Pointer to the next byte, or NULL if no value vas found.

**Parameters**

- `str_p`: Double string.
- `value_p`: Parsed value.

**const** char ***std_strtodfp**(**const** char **str_p*, long **value_p*, int *precision*)
Convert string to decimal fixed point number with given precision.

**Return** Pointer to the next byte, or NULL on failure.

**Parameters**

- `str_p`: Double string.
- `value_p`: Decimal fixed point number of given precision.
- `precision`: Number precision, or decimal places.

int **std_strcpy**(char **dst_p*, far_string_t *src_p*)
Copy string from far memory to memory.

**Return** String length or negative error code.

**Parameters**

- `dst_p`: Normal memory string.
- `src_p`: Far memory string.

int **std_strcmp**(**const** char *_str_p_, far_string_t _fstr_p_)
Compare a string with a far string.

**Return** zero(0) if match, otherwise the difference of the mismatched characters

**Parameters**

- `str_p`: Normal memory string.
- `fstr_p`: Far memory string.

int **std_strcmp_f**(far_string_t _fstr0_p_, far_string_t _fstr1_p_)
Compare two far strings.

**Return** zero(0) if match, otherwise the difference of the mismatched characters.

**Parameters**

- `fstr0_p`: Far memory string.
- `fstr1_p`: Far memory string.

int **std_strncmp**(far_string_t _fstr_p_, **const** char *_str_p_, size_t _size_)
Compare at most `size` bytes of one far string and one string.

**Return** zero(0) if match, otherwise the difference of the mismatched characters.

**Parameters**

- `fstr_p`: Far memory string.
- `str_p`: String.
- `size`: Compare at most size number of bytes.

int **std_strncmp_f**(far_string_t _fstr0_p_, far_string_t _fstr1_p_, size_t _size_)
Compare at most `size` bytes of two far strings.

**Return** zero(0) if match, otherwise the difference of the mismatched characters.

**Parameters**

- `fstr0_p`: Far memory string.
- `fstr1_p`: Far memory string.
- `size`: Compare at most size number of bytes.

int **std_strlen**(far_string_t _fstr_p_)
Get the length in bytes of given far string, not including null termination.

**Return** String length in number of bytes (not including the null termination).

**Parameters**

- `fstr_p`: Far memory string.

char *__std_strip__(char **str_p*, **const** char **strip_p*)
Strip leading and trailing characters from a string. The characters to strip are given by `strip_p`.

**Return** Pointer to the stripped string.

**Parameters**

- `str_p`: String to strip characters from.
- `strip_p`: Characters to strip or NULL for whitespace characters. Must be null-terminated.

ssize_t __std_hexdump__(void **chan_p*, **const** void **buf_p*, size_t *size*)
Write a hex dump of given data to given channel.

**Return** Number of characters written to given channel, or negative error code.

**Parameters**

- `chan_p`: Channel to write the hexdump to.
- `buf_p`: Buffer to dump.
- `size`: Size of buffer.

### 1.6.11 encode

In computing, a character encoding is used to represent a repertoire of characters by some kind of an encoding system.

The encode package on Github.

#### `base64` — Base64 encoding and decoding

Source code: src/encode/base64.h, src/encode/base64.c

Test code: tst/encode/base64/main.c

Test coverage: src/encode/base64.c

---

#### Functions

int __base64_encode__(char **dst_p*, **const** void **src_p*, size_t *size*)
Encode given buffer. The encoded data will be ~33.3% larger than the source data. Choose the destination buffer size accordingly.

**Return** zero(0) or negative error code.

**Parameters**

- `dst_p`: Encoded output data.
- `src_p`: Input data.
- `size`: Number of bytes in the input data.

int **base64_decode**(void *_dst_p_, **const** char *_src_p_, size_t _size_)

Decode given base64 encoded buffer. The decoded data will be ~25% smaller than the destination data. Choose the destination buffer size accordingly.

**Return** zero(0) or negative error code.

**Parameters**

- `dst_p`: Output data.
- `src_p`: Encoded input data.
- `size`: Number of bytes in the encoded input data.

### `json` — JSON encoding and decoding

Source code: src/encode/json.h, src/encode/json.c

Test code: tst/encode/json/main.c

Test coverage: src/encode/json.c

---

### Enums

**enum json_type_t**

JSON type identifier.

*Values:*

**JSON_UNDEFINED** = 0
Undefined type.

**JSON_OBJECT** = 1
Object, `{}`.

**JSON_ARRAY** = 2
Array, `[]`.

**JSON_STRING** = 3
String, `\"...\"`.

**JSON_PRIMITIVE** = 4
Other primitive: number, boolean (true/false) or null.

**enum json_err_t**

*Values:*

**JSON_ERROR_NOMEM** = -1
Not enough tokens were provided.

**JSON_ERROR_INVAL** = -2
Invalid character inside JSON string.

**JSON_ERROR_PART** = -3
The string is not a full JSON packet, more bytes expected.

### Functions

int **json_init**(**struct** *json_t* *self_p, **struct** *json_tok_t* *tokens_p, int *num_tokens*)
Initialize given JSON object. The JSON object must be initialized before it can be used to parse and dump JSON data.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: JSON object to initialize.
- `tokens_p`: Array of tokens. The tokens are either filled by the parsing function `json_parse()`, or already filled by the user when calling this function. The latter can be used to dump the tokens as a string by calling `json_dump()` or `json_dumps()`.
- `num_tokens`: Number of tokens in the array.

int **json_parse**(**struct** *json_t* *self_p, **const** char *js_p, size_t *len*)
Parse given JSON data string into and array of tokens, each describing a single JSON object.

**Return** Number of decoded tokens or negative error code.

**Parameters**

- `self_p`: JSON object.
- `js_p`: JSON string to parse.
- `len`: JSON string length in bytes.

ssize_t **json_dumps**(**struct** *json_t* *self_p, **struct** *json_tok_t* *tokens_p, char *js_p)
Format and write given JSON tokens into a string.

**Return** Dumped string length (not including termination) or negative error code.

**Parameters**

- `self_p`: JSON object.
- `tokens_p`: Root token to dump. Set to NULL to dump the whole object.
- `js_p`: Dumped null terminated JSON string.

ssize_t **json_dump**(**struct** *json_t* *self_p, **struct** *json_tok_t* *tokens_p, void *out_p)
Format and write given JSON tokens to given channel.

**Return** Dumped string length (not including termination) or negative error code.

**Parameters**

- `self_p`: JSON object.
- `tokens_p`: Root token to dump. Set to NULL to dump the whole object.
- `out_p`: Channel to dump the null terminated JSON string to.

**struct** *json_tok_t* ***json_root**(**struct** *json_t* *self_p)
Get the root token.

**Return** The root token or NULL on failure.

**Parameters**

- `self_p`: JSON object.

**struct** *json_tok_t* ***json_object_get**(**struct** *json_t* **self_p*, **const** char **key_p*, **struct** *json_tok_t* **object_p*)

Get the value the string token with given key.

**Return** Token or NULL on error.

**Parameters**

- `self_p`: JSON object.
- `key_p`: Key of the value to get.
- `object_p`: The object to get the value from.

**struct** *json_tok_t* ***json_object_get_primitive**(**struct** *json_t* **self_p*, **const** char **key_p*, **struct** *json_tok_t* **object_p*)

Get the value of the primitive token with given key.

**Return** Token or NULL on error.

**Parameters**

- `self_p`: JSON object.
- `key_p`: Key of the value to get.
- `object_p`: The object to get the value from.

**struct** *json_tok_t* ***json_array_get**(**struct** *json_t* **self_p*, int *index*, **struct** *json_tok_t* **array_p*)

Get the token of given array index.

**Return** Token or NULL on error.

**Parameters**

- `self_p`: JSON object.
- `index`: Index to get.
- `array_p`: The array to get the element from.

void **json_token_object**(**struct** *json_tok_t* **token_p*, int *num_keys*)

Initialize a JSON object token.

**Parameters**

- `token_p`: Initialized token.
- `num_keys`: Number of keys in the object.

void **json_token_array**(**struct** *json_tok_t* **token_p*, int *num_elements*)

Initialize a JSON array token.

**Parameters**

- `token_p`: Initialized token.
- `num_elements`: Number of array elements.

void **json_token_true**(**struct** *json_tok_t* *token_p)
Initialize a JSON boolean true token.

**Parameters**

- `token_p`: Initialized token.

void **json_token_false**(**struct** *json_tok_t* *token_p)
Initialize a JSON boolean false token.

**Parameters**

- `token_p`: Initialized token.

void **json_token_null**(**struct** *json_tok_t* *token_p)
Initialize a JSON null token.

**Parameters**

- `token_p`: Initialized token.

void **json_token_number**(**struct** *json_tok_t* *token_p, **const** char *buf_p, size_t *size*)
Initialize a JSON number (integer/float) token.

**Parameters**

- `token_p`: Initialized token.
- `buf_p`: Number as a string.
- `size`: String length.

void **json_token_string**(**struct** *json_tok_t* *token_p, **const** char *buf_p, size_t *size*)
Initialize a JSON string token.

**Parameters**

- `token_p`: Initialized token.
- `buf_p`: String.
- `size`: String length.

**struct json_tok_t**

### Public Members

*json_type_t* **type**

**const** char ***buf_p**

size_t **size**

int **num_tokens**

**struct json_t**

### Public Members

unsigned int **pos**
Offset in the JSON string.

unsigned int **toknext**
Next token to allocate.

int **toksuper**
Superior token node, e.g parent object or array.

**struct** *json_tok_t* ***tokens_p**
Array of tokens.

int **num_tokens**
Number of tokens in the tokens array.

## nmea — NMEA encoding and decoding

Source code: src/encode/nmea.h, src/encode/nmea.c

Test code: tst/encode/nmea/main.c

Test coverage: src/encode/nmea.c

---

### Defines

**NMEA_SENTENCE_SIZE_MAX**

**NMEA_KNOTS_TO_METERS_PER_SECOND** (knots)

### Enums

**enum nmea_sentence_type_t**
Sentence types.

*Values:*

**nmea_sentence_type_raw_t** = 0

**nmea_sentence_type_gga_t**

**nmea_sentence_type_gll_t**

**nmea_sentence_type_gsa_t**

**nmea_sentence_type_gsv_t**

**nmea_sentence_type_rmc_t**

**nmea_sentence_type_vtg_t**

**nmea_sentence_type_max_t**

### Functions

ssize_t **nmea_encode**(char *_dst_p_, **struct** _nmea_sentence_t_ *_src_p_, size_t _size_)
Encode given NMEA sentence into given buffer.

**Return** Number of bytes in the encoded string, not including the null-termination, or negative error code.

**Parameters**

- `dst_p`: Null-terminated encoded sentence.
- `src_p`: Sentence to encode.
- `size`: Size of the destination buffer `dst_p`.

ssize_t **nmea_decode**(**struct** _nmea_sentence_t_ *_dst_p_, char *_src_p_, size_t _size_)
Decode given NMEA sentence into given struct.

**Return** zero(0) or negative error code.

**Parameters**

- `dst_p`: Decoded NMEA sentence.
- `src_p`: Sentence to decode, starting with a dollar sign and ending with <CR><LF>. This string is modified by this function and the decoded sentence has references to it.
- `size`: Number of bytes in the sentence to decode, not including the null-termination.

int **nmea_decode_fix_time**(char *_src_p_, int *_hour_p_, int *_minute_p_, int *_second_p_)
Decode given NMEA fix time `hhmmss`. The output variables have not been modified if the decoding failed.

**Return** zero(0) or negative error code.

**Parameters**

- `src_p`: Fix time to decode.
- `hour_p`: Decoded hour.
- `minute_p`: Decoded minute.
- `second_p`: Decoded second.

int **nmea_decode_date**(char *_src_p_, int *_year_p_, int *_month_p_, int *_date_p_)
Decode given NMEA date `ddmmyy`. The output variables have not been modified if the decoding failed.

**Return** zero(0) or negative error code.

**Parameters**

- `src_p`: Date to decode.
- `year_p`: Decoded year.
- `month_p`: Decoded month.
- `date_p`: Decoded date.

int **nmea_decode_position**(**struct** *nmea_position_t* *src_p, long *degrees_p)

Decode given NMEA position angle `d{2,3}mm.m+` and direction `[NSEW]`, for example `4703.324 N`, which is decoded as 47 degrees, 3.324 minutes, or 47055400 microdegrees.

The output variable has not been modified if the decoding failed.

**Return** zero(0) or negative error code.

**Parameters**

- `src_p`: Position to decode.
- `degrees_p`: Decoded position in microdegrees (millions of degrees).

**struct nmea_position_t**

#### Public Members

char ***angle_p**

char ***direction_p**

**struct nmea_value_t**

#### Public Members

char ***value_p**

char ***unit_p**

**struct nmea_track_made_good_t**

#### Public Members

char ***value_p**

char ***relative_to_p**

**struct nmea_sentence_raw_t**

*#include <nmea.h>* Raw string data.

#### Public Members

char ***str_p**

**struct nmea_sentence_gga_t**

*#include <nmea.h>* Fix information.

#### Public Members

char ***time_of_fix_p**

**struct** *nmea_position_t* **latitude**

**struct** *nmea_position_t* **longitude**

char *fix_quality_p

char *number_of_tracked_satellites_p

char *horizontal_dilution_of_position_p

struct nmea_value_t altitude

struct nmea_value_t height_of_geoid

struct nmea_sentence_gll_t

#include <nmea.h> Latitude and longitude data.

### Public Members

char *time_of_fix_p

struct nmea_position_t latitude

struct nmea_position_t longitude

char *data_active_p

struct nmea_sentence_gsa_t

#include <nmea.h> Overall satellite data.

### Public Members

char *selection_p

char *fix_p

char *prns[12]

char *pdop_p

char *hdop_p

char *vdop_p

struct nmea_sentence_gsv_t

#include <nmea.h> Detailed satellite data.

### Public Members

char *number_of_sentences_p

char *sentence_p

char *number_of_satellites_p

char *prn_p

char *elevation_p

char *azimuth_p

char *snr_p

struct nmea_sentence_gsv_t::@45 nmea_sentence_gsv_t::satellites[4]

struct nmea_sentence_rmc_t

#include <nmea.h> Recommended minimum data for GPS.

#### Public Members

char ***time_of_fix_p**

char ***status_p**

**struct** *nmea_position_t* **latitude**

**struct** *nmea_position_t* **longitude**

char ***speed_knots_p**

char ***track_angle_p**

char ***date_p**

**struct** *nmea_position_t* **magnetic_variation**

**struct nmea_sentence_vtg_t**

*#include* <*nmea.h*> Track made good and speed over ground.

#### Public Members

**struct** *nmea_track_made_good_t* **track_made_good_true**

**struct** *nmea_track_made_good_t* **track_made_good_magnetic**

**struct** *nmea_value_t* **ground_speed_knots**

**struct** *nmea_value_t* **ground_speed_kmph**

**struct nmea_sentence_t**

*#include* <*nmea.h*> A union of all sentences.

#### Public Members

*nmea_sentence_type_t* **type**

**struct** *nmea_sentence_raw_t* **raw**

**struct** *nmea_sentence_gga_t* **gga**

**struct** *nmea_sentence_gll_t* **gll**

**struct** *nmea_sentence_gsa_t* **gsa**

**struct** *nmea_sentence_gsv_t* **gsv**

**struct** *nmea_sentence_rmc_t* **rmc**

**struct** *nmea_sentence_vtg_t* **vtg**

**union nmea_sentence_t::@46 nmea_sentence_t::@47**

### 1.6.12 hash

A hash function is any function that can be used to map data of arbitrary size to data of fixed size.

The hash package on Github.

### crc — Cyclic Redundancy Checks

Source code: src/hash/crc.h, src/hash/crc.c

Test code: tst/hash/crc/main.c

Test coverage: src/hash/crc.c

---

## Defines

**CRC_8_POLYNOMIAL_8_5_4_0**
The polynomial `x^8 + x^5 + x^4 + x^0`.

## Functions

uint32_t **crc_32**(uint32_t *crc*, **const** void **buf_p*, size_t *size*)
Calculate a 32 bits crc using the polynomial `x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x`

**Return** Calculated crc.

**Parameters**

- `crc`: Initial crc. Often 0x00000000.
- `buf_p`: Buffer to calculate crc of.
- `size`: Size of the buffer.

uint16_t **crc_ccitt**(uint16_t *crc*, **const** void **buf_p*, size_t *size*)
Calculate a 16 bits crc using the CCITT algorithm (polynomial `x^16+x^12+x^5+x^1`).

**Return** Calculated crc.

**Parameters**

- `crc`: Initial crc. Should be 0xffff for CCITT.
- `buf_p`: Buffer to calculate crc of.
- `size`: Size of the buffer.

uint16_t **crc_xmodem**(uint16_t *crc*, **const** void **buf_p*, size_t *size*)
Calculate a 16 bits crc using the XModem algorithm (polynomial `x^16+x^12+x^5+x^1`).

**Return** Calculated crc.

**Parameters**

- `crc`: Initial crc. Should be 0x0000 for XModem.
- `buf_p`: Buffer to calculate crc of.
- `size`: Size of the buffer.

uint8_t **crc_7**(**const** void **buf_p*, size_t *size*)
Calculate a 8 bits crc using the CRC-7 algorithm (polynomial `x^7+x^3+1`).

**Return** Calculated crc.

**Parameters**

- `buf_p`: Buffer to calculate crc of.
- `size`: Size of the buffer.

uint8_t **crc_8**(uint8_t *crc*, uint8_t *polynomial*, **const** void **buf_p*, size_t *size*)
Calculate a 8 bits crc using given polynomial.

**Return** Calculated crc.

**Parameters**

- `crc`: Initial crc. Must be 0x00 on first call.
- `polynimial`: CRC polynomial.
- `buf_p`: Buffer to calculate crc of.
- `size`: Size of the buffer.

### sha1 — SHA1

Source code: src/hash/sha1.h, src/hash/sha1.c

Test code: tst/hash/main.c

Test coverage: src/hash/sha1.c

---

### Functions

int **sha1_init**(**struct** *sha1_t *self_p*)
Initialize given SHA1 object.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SHA1 object.

int **sha1_update**(**struct** *sha1_t *self_p*, void **buf_p*, size_t *size*)
Update the sha object with the given buffer. Repeated calls are equivalent to a single call with the concatenation of all the arguments.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SHA1 object.
- `buf_p`: Buffer to update the sha object with.
- `size`: Size of the buffer.

int **sha1_digest**(**struct** *sha1_t *self_p*, uint8_t **hash_p*)
Return the digest of the strings passed to the sha1_update() method so far. This is a 20-byte value which may contain non-ASCII characters, including null bytes.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: SHA1 object.
- `hash_p`: Hash sum.

**struct sha1_t**

#### Public Members

uint8_t **buf**[64]

uint32_t **size**

**struct sha1_t::@55 sha1_t::block**

uint32_t **h**[5]

uint64_t **size**

### 1.6.13 multimedia

The multimedia package on Github.

#### `midi` — Musical Instrument Digital Interface

Source code: src/multimedia/midi.h, src/multimedia/midi.c

Test code: tst/multimedia/midi/main.c

Test coverage: src/multimedia/midi.c

---

#### Defines

**MIDI_BAUDRATE**

**MIDI_NOTE_OFF**

**MIDI_NOTE_ON**

**MIDI_POLYPHONIC_KEY_PRESSURE**

**MIDI_CONTROL_CHANGE**

**MIDI_PROGRAM_CHANGE**

**MIDI_CHANNEL_PRESSURE**

**MIDI_PITCH_BEND_CHANGE**

**MIDI_SET_INTRUMENT**

**MIDI_PERC**

**MIDI_NOTE_MAX**

**MIDI_NOTE_A0**

MIDI_NOTE_B0

MIDI_NOTE_C1

MIDI_NOTE_D1

MIDI_NOTE_E1

MIDI_NOTE_F1

MIDI_NOTE_G1

MIDI_NOTE_A1

MIDI_NOTE_B1

MIDI_NOTE_C2

MIDI_NOTE_D2

MIDI_NOTE_E2

MIDI_NOTE_F2

MIDI_NOTE_G2

MIDI_NOTE_A2

MIDI_NOTE_B2

MIDI_NOTE_C3

MIDI_NOTE_D3

MIDI_NOTE_E3

MIDI_NOTE_F3

MIDI_NOTE_G3

MIDI_NOTE_A3

MIDI_NOTE_B3

MIDI_NOTE_C4

MIDI_NOTE_D4

MIDI_NOTE_E4

MIDI_NOTE_F4

MIDI_NOTE_G4

MIDI_NOTE_A4

MIDI_NOTE_B4

MIDI_NOTE_C5

MIDI_NOTE_D5

MIDI_NOTE_E5

MIDI_NOTE_F5

MIDI_NOTE_G5

MIDI_NOTE_A5

MIDI_NOTE_B5

MIDI_NOTE_C6

MIDI_NOTE_D6

MIDI_NOTE_E6

MIDI_NOTE_F6

MIDI_NOTE_G6

MIDI_NOTE_A6

MIDI_NOTE_B6

MIDI_NOTE_C7

MIDI_NOTE_D7

MIDI_NOTE_E7

MIDI_NOTE_F7

MIDI_NOTE_G7

MIDI_NOTE_A7

MIDI_NOTE_B7

MIDI_NOTE_C8

MIDI_PERC_ACOUSTIC_BASS_DRUM

MIDI_PERC_BASS_DRUM_1

MIDI_PERC_SIDE_STICK

MIDI_PERC_ACOUSTIC_SNARE

MIDI_PERC_HAND_CLAP

MIDI_PERC_ELECTRIC_SNARE

MIDI_PERC_LOW_FLOOR_TOM

MIDI_PERC_CLOSED_HI_HAT

MIDI_PERC_HIGH_FLOOR_TOM

MIDI_PERC_PEDAL_HI_HAT

MIDI_PERC_LOW_TOM

MIDI_PERC_OPEN_HI_HAT

MIDI_PERC_LOW_MID_TOM

MIDI_PERC_HI_MID_TOM

MIDI_PERC_CRASH_CYMBAL_1

MIDI_PERC_HIGH_TOM

MIDI_PERC_RIDE_CYMBAL_1

MIDI_PERC_CHINESE_CYMBAL

MIDI_PERC_RIDE_BELL

MIDI_PERC_TAMBOURINE

MIDI_PERC_SPLASH_CYMBAL

**MIDI_PERC_COWBELL**

**MIDI_PERC_CRASH_CYMBAL_2**

**MIDI_PERC_VIBRASLAP**

**MIDI_PERC_RIDE_CYMBAL_2**

**MIDI_PERC_HI_BONGO**

**MIDI_PERC_LOW_BONGO**

**MIDI_PERC_MUTE_HI_CONGA**

**MIDI_PERC_OPEN_HI_CONGA**

**MIDI_PERC_LOW_CONGA**

**MIDI_PERC_HIGH_TIMBALE**

**MIDI_PERC_LOW_TIMBALE**

**MIDI_PERC_HIGH_AGOGO**

**MIDI_PERC_LOW_AGOGO**

**MIDI_PERC_CABASA**

**MIDI_PERC_MARACAS**

**MIDI_PERC_SHORT_WHISTLE**

**MIDI_PERC_LONG_WHISTLE**

**MIDI_PERC_SHORT_GUIRO**

**MIDI_PERC_LONG_GUIRO**

**MIDI_PERC_CLAVES**

**MIDI_PERC_HI_WOOD_BLOCK**

**MIDI_PERC_LOW_WOOD_BLOCK**

**MIDI_PERC_MUTE_CUICA**

**MIDI_PERC_OPEN_CUICA**

**MIDI_PERC_MUTE_TRIANGLE**

**MIDI_PERC_OPEN_TRIANGLE**

### Functions

float **midi_note_to_frequency** (int *note*)
Get the frequency for given note.

**Return** Note frequency.

**Parameters**

- `note`: MIDI note.

### 1.6.14 science

Natural sciences helper functions and definitions.

The science package on Github.

#### `math` — Math functions and definitions

Source code: src/science/math.h, src/science/math.c

Test code: tst/science/math/main.c

Test coverage: src/science/math.c

---

#### Defines

**`MATH_PI`**

#### Functions

float **`math_radians_to_degrees`**(float *value*)
Convert given angle from radians to degrees.

**Return** Angle in degrees, or NaN if an error occurred.

**Parameters**

- `angle`: Angle in radians.

float **`math_degrees_to_radians`**(float *value*)
Convert given angle from degrees to radians.

**Return** Angle in radians, or NaN if an error occurred.

**Parameters**

- `angle`: Angle in degrees.

int32_t **`math_log2_fixed_point`**(uint32_t *x*, int *precision*)
Calculate the 2-logarithm of given value in given fixed point precision.

**Return** 2-logarithm of given value x, in given precision.

**Parameters**

- `x`: Value to calculate the 2-logarithm of.
- `precision`: Fixed point precision in the range 1 to 31, inclusive.

int32_t **`math_ln_fixed_point`**(uint32_t *x*, int *precision*)
Calculate the natural logarithm of given value in given fixed point precision.

**Return** Natural logarithm of given value x, in given precision.

**Parameters**

- `x`: Value to calculate the natural logarithm of.
- `precision`: Fixed point precision in the range 1 to 31, inclusive.

int32_t **math_log10_fixed_point**(uint32_t *x*, int *precision*)
Calculate the 10-logarithm of given value in given fixed point precision.

**Return** 10-logarithm of given value x, in given precision.

**Parameters**

- `x`: Value to calculate the 10-logarithm of.
- `precision`: Fixed point precision in the range 1 to 31, inclusive.

### `science` — Common science functions and definitions

Source code: src/science/science.h, src/science/science.c

Test code: tst/science/science/main.c

Test coverage: src/science/science.c

---

#### Defines

**SCIENCE_SEA_LEVEL_STANDARD_PRESSURE**

#### Functions

int **science_module_init**(void)
Initialize the science module. This function must be called before calling any other function in this module.

The module will only be initialized once even if this function is called multiple times.

**Return** zero(0) or negative error code.

float **science_pressure_to_altitude**(float *pressure*, float *pressure_at_sea_level*)
Convert given pressure to an its altitude.

**Return** Altitude in meters, or NaN if an error occurred.

**Parameters**

- `pressure`: Pressure in Pascal.
- `pressure_at_sea_level`: Sea level pressure in Pascal.

float **science_pressure_from_altitude**(float *altitude*, float *pressure_at_sea_level*)
Convert given altitude to an its pressure.

**Return** Pressure in Pascal, or NaN if an error occurred.

**Parameters**

- `altitude`: Altitude in meters.

- `pressure_at_sea_level`: Sea level pressure in Pascal.

float **science_mps_to_kmph**(float *speed*)
Convert given speed from m/s to km/h.

**Return** Speed in km/h, or NaN if an error occurred.

**Parameters**

- `speed`: Speed in m/s.

float **science_mps_from_kmph**(float *speed*)
Convert given speed from km/h to m/s.

**Return** Speed in m/s, or NaN if an error occurred.

**Parameters**

- `speed`: Speed in km/h.

float **science_mps_to_knots**(float *speed*)
Convert given speed from m/s to knots.

**Return** Speed in knots, or NaN if an error occurred.

**Parameters**

- `speed`: Speed in m/s.

float **science_mps_from_knots**(float *speed*)
Convert given speed from knots to m/s.

**Return** Speed in m/s, or NaN if an error occurred.

**Parameters**

- `speed`: Speed in knots.

float **science_mps_to_mph**(float *speed*)
Convert given speed from m/s to mi/h.

**Return** Speed in mi/h, or NaN if an error occurred.

**Parameters**

- `speed`: Speed in m/s.

float **science_mps_from_mph**(float *speed*)
Convert given speed from mi/h to m/s.

**Return** Speed in m/s, or NaN if an error occurred.

**Parameters**

- `speed`: Speed in mi/h.

### 1.6.15 boards

The boards supported by *Simba*.

The boards on Github.

#### arduino_due — Arduino Due

Source code: src/boards/arduino_due/board.h, src/boards/arduino_due/board.c

Hardware reference: *Arduino Due*

---

#### Defines

**pin_d0_dev**

**pin_d1_dev**

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**pin_d9_dev**

**pin_d10_dev**

**pin_d11_dev**

**pin_d12_dev**

**pin_d13_dev**

**pin_d14_dev**

**pin_d15_dev**

**pin_d16_dev**

**pin_d17_dev**

**pin_d18_dev**

**pin_d19_dev**

**pin_d20_dev**

**pin_d21_dev**

**pin_d22_dev**

**pin_d23_dev**

**pin_d24_dev**

**pin_d25_dev**

**pin_d26_dev**

**pin_d27_dev**

**pin_d28_dev**

**pin_d29_dev**

**pin_d30_dev**

**pin_d31_dev**

**pin_d32_dev**

**pin_d33_dev**

**pin_d34_dev**

**pin_d35_dev**

**pin_d36_dev**

**pin_d37_dev**

**pin_d38_dev**

**pin_d39_dev**

**pin_d40_dev**

**pin_d41_dev**

**pin_d42_dev**

**pin_d43_dev**

**pin_d44_dev**

**pin_d45_dev**

**pin_d46_dev**

**pin_d47_dev**

**pin_d48_dev**

**pin_d49_dev**

**pin_d50_dev**

**pin_d51_dev**

**pin_d52_dev**

**pin_d53_dev**

**pin_a0_dev**

**pin_a1_dev**

**pin_a2_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_a6_dev**

```
pin_a7_dev
pin_a8_dev
pin_a9_dev
pin_a10_dev
pin_a11_dev
pin_led_dev
pin_dac0_dev
pin_dac1_dev
exti_d0_dev
exti_d1_dev
exti_d2_dev
exti_d3_dev
exti_d4_dev
exti_d5_dev
exti_d6_dev
exti_d7_dev
exti_d8_dev
exti_d9_dev
exti_d10_dev
exti_d11_dev
exti_d12_dev
exti_d13_dev
exti_d14_dev
exti_d15_dev
exti_d16_dev
exti_d17_dev
exti_d18_dev
exti_d19_dev
exti_d20_dev
exti_d21_dev
exti_d22_dev
exti_d23_dev
exti_d24_dev
exti_d25_dev
exti_d26_dev
exti_d27_dev
```

**exti_d28_dev**

**exti_d29_dev**

**exti_d30_dev**

**exti_d31_dev**

**exti_d32_dev**

**exti_d33_dev**

**exti_d34_dev**

**exti_d35_dev**

**exti_d36_dev**

**exti_d37_dev**

**exti_d38_dev**

**exti_d39_dev**

**exti_d40_dev**

**exti_d41_dev**

**exti_d42_dev**

**exti_d43_dev**

**exti_d44_dev**

**exti_d45_dev**

**exti_d46_dev**

**exti_d47_dev**

**exti_d48_dev**

**exti_d49_dev**

**exti_d50_dev**

**exti_d51_dev**

**exti_d52_dev**

**exti_d53_dev**

**exti_a0_dev**

**exti_a1_dev**

**exti_a2_dev**

**exti_a3_dev**

**exti_a4_dev**

**exti_a5_dev**

**exti_a6_dev**

**exti_a7_dev**

**exti_a8_dev**

**exti_a9_dev**

**exti_a10_dev**

**exti_a11_dev**

**exti_led_dev**

**exti_dac0_dev**

**exti_dac1_dev**

**pwm_d2_dev**

**pwm_d3_dev**

**pwm_d5_dev**

**pwm_d6_dev**

**pwm_d7_dev**

**pwm_d8_dev**

**pwm_d9_dev**

**pwm_d10_dev**

**pwm_d11_dev**

**pwm_d12_dev**

**adc_0_dev**

**dac_0_dev**

**flash_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *_str_p_)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `arduino_mega` — Arduino Mega

Source code: src/boards/arduino_mega/board.h, src/boards/arduino_mega/board.c

Hardware reference: *Arduino Mega*

---

### Defines

**pin_d0_dev**

**pin_d1_dev**

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**pin_d9_dev**

**pin_d10_dev**

**pin_d11_dev**

**pin_d12_dev**

**pin_d13_dev**

**pin_d14_dev**

**pin_d15_dev**

**pin_d16_dev**

**pin_d17_dev**

**pin_d18_dev**

**pin_d19_dev**

**pin_d20_dev**

**pin_d21_dev**

**pin_d22_dev**

**pin_d23_dev**

**pin_d24_dev**

**pin_d25_dev**

**pin_d26_dev**

**pin_d27_dev**

**pin_d28_dev**

**pin_d29_dev**

**pin_d30_dev**

**pin_d31_dev**

**pin_d32_dev**

**pin_d33_dev**

**pin_d34_dev**

**pin_d35_dev**

**pin_d36_dev**

**pin_d37_dev**

**pin_d38_dev**

```
pin_d39_dev
pin_d40_dev
pin_d41_dev
pin_d42_dev
pin_d43_dev
pin_d44_dev
pin_d45_dev
pin_d46_dev
pin_d47_dev
pin_d48_dev
pin_d49_dev
pin_d50_dev
pin_d51_dev
pin_d52_dev
pin_d53_dev
pin_a0_dev
pin_a1_dev
pin_a2_dev
pin_a3_dev
pin_a4_dev
pin_a5_dev
pin_a6_dev
pin_a7_dev
pin_a8_dev
pin_a9_dev
pin_a10_dev
pin_a11_dev
pin_a12_dev
pin_a13_dev
pin_a14_dev
pin_a15_dev
pin_led_dev
exti_d2_dev
exti_d3_dev
exti_d18_dev
exti_d19_dev
```

**exti_d20_dev**

**exti_d21_dev**

**pcint_d10_dev**

**pcint_d11_dev**

**pcint_d12_dev**

**pcint_d13_dev**

**pcint_d14_dev**

**pcint_d15_dev**

**pcint_d50_dev**

**pcint_d51_dev**

**pcint_d52_dev**

**pcint_d53_dev**

**pcint_a8_dev**

**pcint_a9_dev**

**pcint_a10_dev**

**pcint_a11_dev**

**pcint_a12_dev**

**pcint_a13_dev**

**pcint_a14_dev**

**pcint_a15_dev**

**pwm_d2_dev**

**pwm_d3_dev**

**pwm_d4_dev**

**pwm_d5_dev**

**pwm_d6_dev**

**pwm_d7_dev**

**pwm_d8_dev**

**pwm_d9_dev**

**pwm_d10_dev**

**pwm_d13_dev**

**adc_0_dev**

**i2c_0_dev**

**spi_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `arduino_nano` — Arduino Nano

Source code: src/boards/arduino_nano/board.h, src/boards/arduino_nano/board.c

Hardware reference: *Arduino Nano*

---

### Defines

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**pin_d9_dev**

**pin_d10_dev**

**pin_d11_dev**

**pin_d12_dev**

**pin_d13_dev**

**pin_a0_dev**

**pin_a1_dev**

**pin_a2_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_led_dev**

**exti_d2_dev**

**exti_d3_dev**

**pcint_d2_dev**

**pcint_d3_dev**

**pcint_d4_dev**

**pcint_d5_dev**

**pcint_d6_dev**

**pcint_d7_dev**

**pcint_d8_dev**

**pcint_d9_dev**

**pcint_d10_dev**

**pcint_d11_dev**

**pcint_d12_dev**

**pcint_d13_dev**

**pcint_a0_dev**

**pcint_a1_dev**

**pcint_a2_dev**

**pcint_a3_dev**

**pcint_a4_dev**

**pcint_a5_dev**

**pwm_d3_dev**

**pwm_d5_dev**

**pwm_d6_dev**

**pwm_d11_dev**

**adc_0_dev**

**i2c_0_dev**

**spi_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *_str_p_)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `arduino_pro_micro` — Arduino Pro Micro

Source code: src/boards/arduino_pro_micro/board.h, src/boards/arduino_pro_micro/board.c

Hardware reference: *Arduino Pro Micro*

### Defines

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**pin_d9_dev**

**pin_d10_dev**

**pin_d14_dev**

**pin_d15_dev**

**pin_d16_dev**

**pin_a0_dev**

**pin_a1_dev**

**pin_a2_dev**

**pin_a3_dev**

**pin_led_dev**

**exti_d2_dev**

**exti_d3_dev**

**pwm_d3_dev**

**pwm_d9_dev**

**pwm_d10_dev**

**pwm_d11_dev**

**adc_0_dev**

**i2c_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *_str_p_)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

### arduino_uno — Arduino Uno

Source code: src/boards/arduino_uno/board.h, src/boards/arduino_uno/board.c

Hardware reference: *Arduino Uno*

---

#### Defines

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**pin_d9_dev**

**pin_d10_dev**

**pin_d11_dev**

**pin_d12_dev**

**pin_d13_dev**

**pin_a0_dev**

**pin_a1_dev**

**pin_a2_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_led_dev**

**exti_d2_dev**

**exti_d3_dev**

**pcint_d2_dev**

**pcint_d3_dev**

**pcint_d4_dev**

**pcint_d5_dev**

**pcint_d6_dev**

**pcint_d7_dev**

**pcint_d8_dev**

**pcint_d9_dev**

**pcint_d10_dev**

**pcint_d11_dev**

**pcint_d12_dev**

**pcint_d13_dev**

**pcint_a0_dev**

**pcint_a1_dev**

**pcint_a2_dev**

**pcint_a3_dev**

**pcint_a4_dev**

**pcint_a5_dev**

**pwm_d3_dev**

**pwm_d5_dev**

**pwm_d6_dev**

**pwm_d11_dev**

**adc_0_dev**

**i2c_0_dev**

**spi_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `defcon26_badge` — DEF CON 26 Badge

Source code: src/boards/defcon26_badge/board.h, src/boards/defcon26_badge/board.c

Hardware reference: *DEF CON 26 Badge*

---

### Defines

**pin_ra0_dev**

**pin_ra1_dev**

**pin_ra2_dev**

**pin_ra3_dev**

**pin_ra4_dev**

**pin_ra6_dev**

**pin_ra7_dev**

**pin_ra8_dev**

**pin_ra9_dev**

**pin_ra10_dev**

**pin_ra15_dev**

**pin_rb0_dev**

**pin_rb1_dev**

**pin_rb2_dev**

**pin_rb3_dev**

**pin_rb4_dev**

**pin_rb5_dev**

**pin_rb6_dev**

**pin_rb7_dev**

**pin_rb8_dev**

**pin_rb9_dev**

**pin_rb10_dev**

**pin_rb11_dev**

**pin_rb12_dev**

**pin_rb13_dev**

**pin_rb14_dev**

**pin_rb15_dev**

**pin_rc0_dev**

**pin_rc1_dev**

**pin_rc2_dev**

**pin_rc3_dev**

**pin_rc4_dev**

**pin_rc5_dev**

**pin_rc6_dev**

**pin_rc7_dev**

**pin_rc8_dev**

**pin_rc9_dev**

**pin_rc12_dev**

**pin_rd0_dev**

**pin_led_dev**

**flash_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)
Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- str_p: Pin as a string.

## esp01 — ESP8266 Development Board

Source code: src/boards/esp01/board.h, src/boards/esp01/board.c

Hardware reference: *ESP-01*

---

### Defines

**pin_gpio0_dev**

**pin_gpio1_dev**

**pin_gpio2_dev**

**pin_d0_dev**

**pin_d1_dev**

**pin_d2_dev**

**exti_d0_dev**

**exti_d2_dev**

**pin_led_dev**

**flash_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)
Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- str_p: Pin as a string.

### esp12e — ESP8266 Development Board

Source code: src/boards/esp12e/board.h, src/boards/esp12e/board.c

Hardware reference: *ESP-12E Development Board*

---

#### Defines

**pin_gpio0_dev**

**pin_gpio2_dev**

**pin_gpio4_dev**

**pin_gpio5_dev**

**pin_gpio12_dev**

**pin_gpio13_dev**

**pin_gpio14_dev**

**pin_gpio15_dev**

**pin_gpio16_dev**

**pin_d0_dev**

**pin_d2_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d12_dev**

**pin_d13_dev**

**pin_d14_dev**

**pin_d15_dev**

**pin_d16_dev**

**exti_d0_dev**

**exti_d2_dev**

**exti_d4_dev**

**exti_d5_dev**

**exti_d12_dev**

**exti_d13_dev**

**exti_d14_dev**

**exti_d15_dev**

**pin_led_dev**

**pin_a0_dev**

**adc_0_dev**

**flash_0_dev**

**ADC_PINS_MAX**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)
Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## esp32_devkitc — ESP32-DevKitC

Source code: src/boards/esp32_devkitc/board.h, src/boards/esp32_devkitc/board.c

Hardware reference: *ESP32-DevKitC*

---

### Defines

**pin_gpio00_dev**

**pin_gpio01_dev**

**pin_gpio02_dev**

**pin_gpio03_dev**

**pin_gpio04_dev**

**pin_gpio05_dev**

**pin_gpio06_dev**

**pin_gpio07_dev**

**pin_gpio08_dev**

**pin_gpio09_dev**

**pin_gpio10_dev**

**pin_gpio11_dev**

**pin_gpio12_dev**

**pin_gpio13_dev**

**pin_gpio14_dev**

**pin_gpio15_dev**

**pin_gpio16_dev**

**pin_gpio17_dev**

**pin_gpio18_dev**

**pin_gpio19_dev**

**pin_gpio21_dev**

**pin_gpio22_dev**

**pin_gpio23_dev**

**pin_gpio25_dev**

**pin_gpio26_dev**

**pin_gpio27_dev**

**pin_gpio32_dev**

**pin_gpio33_dev**

**pin_gpio34_dev**

**pin_gpio35_dev**

**pin_gpio36_dev**

**pin_gpio39_dev**

**pin_led_dev**

**pin_dac1_dev**

**pin_dac2_dev**

**pin_a0_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_a6_dev**

**pin_a7_dev**

**i2c_dev**

**spi_h_dev**

**spi_v_dev**

**adc_0_dev**

**adc_1_dev**

**flash_0_dev**

**dac_0_dev**

**ADC_PINS_MAX**

### Functions

int **board_pin_string_to_device_index**(**const** char *_str_p_)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

### `huzzah` — Huzzah

Source code: src/boards/huzzah/board.h, src/boards/huzzah/board.c

Hardware reference: *Adafruit HUZZAH ESP8266 breakout*

---

#### Defines

**pin_gpio0_dev**

**pin_gpio2_dev**

**pin_gpio4_dev**

**pin_gpio5_dev**

**pin_gpio12_dev**

**pin_gpio13_dev**

**pin_gpio14_dev**

**pin_gpio15_dev**

**pin_gpio16_dev**

**pin_d0_dev**

**pin_d2_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d12_dev**

**pin_d13_dev**

**pin_d14_dev**

**pin_d15_dev**

**pin_d16_dev**

**pin_led_dev**

**pin_a0_dev**

**adc_0_dev**

**flash_0_dev**

**ADC_PINS_MAX**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `linux` — Linux

Source code: src/boards/linux/board.h, src/boards/linux/board.c

---

### Defines

**PIN_DEVICE_BASE**

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**pin_d9_dev**

**pin_d10_dev**

**pin_d11_dev**

**pin_d12_dev**

**pin_d13_dev**

**pin_a0_dev**

**pin_a1_dev**

**pin_a2_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_a6_dev**

**pin_a7_dev**

**pin_led_dev**

**pwm_d3_dev**

**pwm_d9_dev**

**pwm_d10_dev**

**pwm_d11_dev**

**adc_0_dev**

**pin_dac0_dev**

**pin_dac1_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `maple_esp32` — Maple Esp32

Source code: src/boards/maple_esp32/board.h, src/boards/maple_esp32/board.c

Hardware reference: *Maple-ESP32*

---

### Defines

**pin_gpio00_dev**

**pin_gpio01_dev**

**pin_gpio02_dev**

**pin_gpio03_dev**

**pin_gpio04_dev**

**pin_gpio05_dev**

**pin_gpio06_dev**

**pin_gpio07_dev**

**pin_gpio08_dev**

**pin_gpio09_dev**

**pin_gpio10_dev**

**pin_gpio11_dev**

**pin_gpio12_dev**

**pin_gpio13_dev**

**pin_gpio14_dev**

**pin_gpio15_dev**

**pin_gpio16_dev**

**pin_gpio17_dev**

**pin_gpio18_dev**

**pin_gpio19_dev**

**pin_gpio21_dev**

**pin_gpio22_dev**

**pin_gpio23_dev**

**pin_gpio25_dev**

**pin_gpio26_dev**

**pin_gpio27_dev**

**pin_gpio32_dev**

**pin_gpio33_dev**

**pin_gpio34_dev**

**pin_gpio35_dev**

**pin_gpio36_dev**

**pin_gpio39_dev**

**pin_led_dev**

**pin_dac1_dev**

**pin_dac2_dev**

**pin_a0_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_a6_dev**

**pin_a7_dev**

**i2c_dev**

**spi_h_dev**

**spi_v_dev**

**adc_0_dev**

**adc_1_dev**

**flash_0_dev**

**dac_0_dev**

**ADC_PINS_MAX**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)
Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## nano32 — Nano32

Source code: src/boards/nano32/board.h, src/boards/nano32/board.c

Hardware reference: *Nano32*

---

### Defines

**pin_gpio00_dev**

**pin_gpio01_dev**

**pin_gpio02_dev**

**pin_gpio03_dev**

**pin_gpio04_dev**

**pin_gpio05_dev**

**pin_gpio06_dev**

**pin_gpio07_dev**

**pin_gpio08_dev**

**pin_gpio09_dev**

**pin_gpio10_dev**

**pin_gpio11_dev**

**pin_gpio12_dev**

**pin_gpio13_dev**

**pin_gpio14_dev**

**pin_gpio15_dev**

**pin_gpio16_dev**

**pin_gpio17_dev**

**pin_gpio18_dev**

**pin_gpio19_dev**

**pin_gpio21_dev**

**pin_gpio22_dev**

**pin_gpio23_dev**

**pin_gpio25_dev**

**pin_gpio26_dev**

**pin_gpio27_dev**

**pin_gpio32_dev**

**pin_gpio33_dev**

**pin_gpio34_dev**

**pin_gpio35_dev**

**pin_gpio36_dev**

**pin_gpio39_dev**

**pin_led_dev**

**pin_dac1_dev**

**pin_dac2_dev**

**pin_a0_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_a6_dev**

**pin_a7_dev**

**i2c_dev**

**spi_h_dev**

**spi_v_dev**

**adc_0_dev**

**adc_1_dev**

**flash_0_dev**

**dac_0_dev**

**ADC_PINS_MAX**

### Functions

int **board_pin_string_to_device_index**(**const** char **str_p*)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

### nodemcu — NodeMCU

Source code: src/boards/nodemcu/board.h, src/boards/nodemcu/board.c

Hardware reference: *NodeMCU*

---

#### Defines

**pin_d0_dev**

**pin_d1_dev**

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**pin_d9_dev**

**pin_d10_dev**

**exti_d1_dev**

**exti_d2_dev**

**exti_d3_dev**

**exti_d4_dev**

**exti_d5_dev**

**exti_d6_dev**

**exti_d7_dev**

**exti_d8_dev**

**exti_d9_dev**

**exti_d10_dev**

**pin_led_dev**

**pin_a0_dev**

**adc_0_dev**

**flash_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `nrf52840_pdk` — NRF52840_PDK

Source code: src/boards/nrf52840_pdk/board.h, src/boards/nrf52840_pdk/board.c

---

### Defines

**pin_p00_dev**

**pin_p01_dev**

**pin_p02_dev**

**pin_p03_dev**

**pin_p04_dev**

**pin_p05_dev**

**pin_p06_dev**

**pin_p07_dev**

**pin_p08_dev**

**pin_p09_dev**

**pin_p10_dev**

**pin_p11_dev**

**pin_p12_dev**

**pin_p13_dev**

**pin_p14_dev**

**pin_p15_dev**

**pin_p16_dev**

**pin_p17_dev**

**pin_p18_dev**

**pin_p19_dev**

**pin_p20_dev**

**pin_p21_dev**

**pin_p22_dev**

**pin_p23_dev**

**pin_p24_dev**

**pin_p25_dev**

**pin_p26_dev**

**pin_p27_dev**

**pin_p28_dev**

**pin_p29_dev**

**pin_p30_dev**

**pin_p31_dev**

**pin_ain0_dev**

**pin_ain1_dev**

**pin_ain2_dev**

**pin_ain3_dev**

**pin_ain4_dev**

**pin_ain5_dev**

**pin_ain6_dev**

**pin_ain7_dev**

**pin_btn1_dev**

**pin_btn2_dev**

**pin_btn3_dev**

**pin_btn4_dev**

**pin_led1_dev**

**pin_led2_dev**

**pin_led3_dev**

**pin_led4_dev**

**pin_led_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char **str_p*)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

### photon — Photon

Source code: src/boards/photon/board.h, src/boards/photon/board.c

Hardware reference: *Photon*

---

#### Defines

**pin_d0_dev**

**pin_d1_dev**

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_a0_dev**

**pin_a1_dev**

**pin_a2_dev**

**pin_a3_dev**

**pin_a4_dev**

**pin_a5_dev**

**pin_led_dev**

**pin_dac0_dev**

**pin_dac1_dev**

**pwm_d0_dev**

**pwm_d1_dev**

**pwm_d2_dev**

**pwm_d3_dev**

**pwm_a4_dev**

**pwm_a5_dev**

**flash_0_dev**

#### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

### `spc56ddiscovery` — SPC56D-Discovery

Source code: src/boards/spc56ddiscovery/board.h, src/boards/spc56ddiscovery/board.c

Hardware reference: *SPC56D Discovery*

---

#### Defines

**pin_pa0_dev**

**pin_pa1_dev**

**pin_pa2_dev**

**pin_pa3_dev**

**pin_pa4_dev**

**pin_pa5_dev**

**pin_pa6_dev**

**pin_pa7_dev**

**pin_pa8_dev**

**pin_pa9_dev**

**pin_pa10_dev**

**pin_pa11_dev**

**pin_pa12_dev**

**pin_pa13_dev**

**pin_pa14_dev**

**pin_pa15_dev**

**pin_pb0_dev**

**pin_pb1_dev**

**pin_pb2_dev**

**pin_pb3_dev**

**pin_pb4_dev**

**pin_pb5_dev**

**pin_pb6_dev**

**pin_pb7_dev**

**pin_pb8_dev**

**pin_pb9_dev**

**pin_pb10_dev**

**pin_pb11_dev**

**pin_pb12_dev**

**pin_pb13_dev**

**pin_pb14_dev**

**pin_pb15_dev**

**pin_pc0_dev**

**pin_pc1_dev**

**pin_pc2_dev**

**pin_pc3_dev**

**pin_pc4_dev**

**pin_pc5_dev**

**pin_pc6_dev**

**pin_pc7_dev**

**pin_pc8_dev**

**pin_pc9_dev**

**pin_pc10_dev**

**pin_led_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *_str_p_)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `stm32f3discovery` — STM32F3DISCOVERY

Source code: src/boards/stm32f3discovery/board.h, src/boards/stm32f3discovery/board.c

Hardware reference: *STM32F3DISCOVERY*

---

### Defines

**pin_pa0_dev**

**pin_pa1_dev**

**pin_pa2_dev**

```
pin_pa3_dev
pin_pa4_dev
pin_pa5_dev
pin_pa6_dev
pin_pa7_dev
pin_pa8_dev
pin_pa9_dev
pin_pa10_dev
pin_pa11_dev
pin_pa12_dev
pin_pa13_dev
pin_pa14_dev
pin_pa15_dev
pin_pb0_dev
pin_pb1_dev
pin_pb2_dev
pin_pb3_dev
pin_pb4_dev
pin_pb5_dev
pin_pb6_dev
pin_pb7_dev
pin_pb8_dev
pin_pb9_dev
pin_pb10_dev
pin_pb11_dev
pin_pb12_dev
pin_pb13_dev
pin_pb14_dev
pin_pb15_dev
pin_pc0_dev
pin_pc1_dev
pin_pc2_dev
pin_pc3_dev
pin_pc4_dev
pin_pc5_dev
pin_pc6_dev
```

**pin_pc7_dev**

**pin_pc8_dev**

**pin_pc9_dev**

**pin_pc10_dev**

**pin_pc11_dev**

**pin_pc12_dev**

**pin_pc13_dev**

**pin_pc14_dev**

**pin_pc15_dev**

**pin_pd0_dev**

**pin_pd1_dev**

**pin_pd2_dev**

**pin_pd3_dev**

**pin_pd4_dev**

**pin_pd5_dev**

**pin_pd6_dev**

**pin_pd7_dev**

**pin_pd8_dev**

**pin_pd9_dev**

**pin_pd10_dev**

**pin_pd11_dev**

**pin_pd12_dev**

**pin_pd13_dev**

**pin_pd14_dev**

**pin_pd15_dev**

**pin_pe0_dev**

**pin_pe1_dev**

**pin_pe2_dev**

**pin_pe3_dev**

**pin_pe4_dev**

**pin_pe5_dev**

**pin_pe6_dev**

**pin_pe7_dev**

**pin_pe8_dev**

**pin_pe9_dev**

**pin_pe10_dev**

**pin_pe11_dev**

**pin_pe12_dev**

**pin_pe13_dev**

**pin_pe14_dev**

**pin_pe15_dev**

**uart_0_dev**

**uart_1_dev**

**uart_2_dev**

**spi_0_dev**

**spi_1_dev**

**spi_2_dev**

**i2c_0_dev**

**i2c_1_dev**

**can_0_dev**

**flash_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *_str_p_)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `stm32vldiscovery` — STM32VLDISCOVERY

Source code: src/boards/stm32vldiscovery/board.h, src/boards/stm32vldiscovery/board.c

Hardware reference: *STM32VLDISCOVERY*

---

### Defines

**pin_pa0_dev**

**pin_pa1_dev**

**pin_pa2_dev**

**pin_pa3_dev**

**pin_pa4_dev**

**pin_pa5_dev**

**pin_pa6_dev**

**pin_pa7_dev**

**pin_pa8_dev**

**pin_pa9_dev**

**pin_pa10_dev**

**pin_pa11_dev**

**pin_pa12_dev**

**pin_pa13_dev**

**pin_pa14_dev**

**pin_pa15_dev**

**pin_pb0_dev**

**pin_pb1_dev**

**pin_pb2_dev**

**pin_pb3_dev**

**pin_pb4_dev**

**pin_pb5_dev**

**pin_pb6_dev**

**pin_pb7_dev**

**pin_pb8_dev**

**pin_pb9_dev**

**pin_pb10_dev**

**pin_pb11_dev**

**pin_pb12_dev**

**pin_pb13_dev**

**pin_pb14_dev**

**pin_pb15_dev**

**pin_pc0_dev**

**pin_pc1_dev**

**pin_pc2_dev**

**pin_pc3_dev**

**pin_pc4_dev**

**pin_pc5_dev**

**pin_pc6_dev**

**pin_pc7_dev**

**pin_pc8_dev**

**pin_pc9_dev**

**pin_pc10_dev**

**pin_pc11_dev**

**pin_pc12_dev**

**pin_pc13_dev**

**pin_pc14_dev**

**pin_pc15_dev**

**pin_pd0_dev**

**pin_pd1_dev**

**pin_pd2_dev**

**pin_led_dev**

**pin_ld3_dev**

**pin_ld4_dev**

**uart_0_dev**

**uart_1_dev**

**uart_2_dev**

**spi_0_dev**

**spi_1_dev**

**spi_2_dev**

**i2c_0_dev**

**i2c_1_dev**

**flash_0_dev**

### Functions

int **board_pin_string_to_device_index**(**const** char *_str_p_)

Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- `str_p`: Pin as a string.

## `wemos_d1_mini` — WEMOS D1 Mini

Source code: src/boards/wemos_d1_mini/board.h, src/boards/wemos_d1_mini/board.c

Hardware reference: *WEMOS D1 mini*

### Defines

**pin_gpio0_dev**

**pin_gpio2_dev**

**pin_gpio4_dev**

**pin_gpio5_dev**

**pin_gpio12_dev**

**pin_gpio13_dev**

**pin_gpio14_dev**

**pin_gpio15_dev**

**pin_gpio16_dev**

**pin_d0_dev**

**pin_d1_dev**

**pin_d2_dev**

**pin_d3_dev**

**pin_d4_dev**

**pin_d5_dev**

**pin_d6_dev**

**pin_d7_dev**

**pin_d8_dev**

**exti_d1_dev**

**exti_d2_dev**

**exti_d3_dev**

**exti_d4_dev**

**exti_d5_dev**

**exti_d6_dev**

**exti_d7_dev**

**exti_d8_dev**

**pin_led_dev**

**pin_a0_dev**

**adc_0_dev**

**flash_0_dev**

**ADC_PINS_MAX**

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)
Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- str_p: Pin as a string.

#### xvisor_raspberry_pi_3 — Xvisor Raspberry Pi 3

Source code: src/boards/xvisor_raspberry_pi_3/board.h, src/boards/xvisor_raspberry_pi_3/board.c

Hardware reference: *Xvisor Raspberry Pi 3*

---

### Functions

int **board_pin_string_to_device_index**(**const** char *str_p)
Convert given pin string to the pin number.

**Return** Pin number or negative error code.

**Parameters**

- str_p: Pin as a string.

## 1.6.16 mcus

The Micro Controller Units (MCU:s) supported by *Simba*.

The MCU:s on Github.

#### atmega2560 — ATMega2560

Source code: src/mcus/atmega2560/mcu.h

---

### Defines

**PIN_DEVICE_MAX**

**EXTI_DEVICE_MAX**

**SPI_DEVICE_MAX**

**UART_DEVICE_MAX**

**PWM_DEVICE_MAX**

**ADC_DEVICE_MAX**

**I2C_DEVICE_MAX**

**PCINT_DEVICE_MAX**

### atmega328p — ATMega328p

Source code: src/mcus/atmega328p/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**EXTI_DEVICE_MAX**

**SPI_DEVICE_MAX**

**UART_DEVICE_MAX**

**PWM_DEVICE_MAX**

**ADC_DEVICE_MAX**

**I2C_DEVICE_MAX**

**PCINT_DEVICE_MAX**

**USART0_TX_vect**

**USART0_RX_vect**

**USART0_UDRE_vect**

### atmega32u4 — ATMega32u4

Source code: src/mcus/atmega32u4/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**EXTI_DEVICE_MAX**

**SPI_DEVICE_MAX**

**UART_DEVICE_MAX**

**PWM_DEVICE_MAX**

**ADC_DEVICE_MAX**

**I2C_DEVICE_MAX**

**USB_DEVICE_MAX**

**USART0_TX_vect**

**USART0_RX_vect**

**USART0_UDRE_vect**

**UCSZ00**

**UCSZ01**

**UCSZ02**

**UPM00**

**UPM01**

**USBS0**

**U2X0**

**UPE0**

**DOR0**

**FE0**

**TXC0**

**RXCIE0**

**RXEN0**

**TXEN0**

**UDRE0**

**UDRIE0**

**TXCIE0**

### esp32 — Esp32

### I2C

Simba does not yet support the ESP32 hardware I2C support, but can use the *i2c_soft — Software I2C* driver instead. By default one *i2c — I2C* compatability device is instantiated:

| Device | SCL | SDA |
|---|---|---|
| i2c0 | Pin 22 | Pin 21 |

---

Hardware reference: https://github.com/eerimoq/hardware-reference/tree/master/esp32

Source code: src/mcus/esp32/mcu.h

---

### Defines

**PIN_DEVICE_MAX**

**EXTI_DEVICE_MAX**

**SPI_DEVICE_MAX**

**UART_DEVICE_MAX**

**ADC_DEVICE_MAX**

**I2C_DEVICE_MAX**

**FLASH_DEVICE_MAX**

**CAN_DEVICE_MAX**

**DAC_DEVICE_MAX**

### esp8266 — Esp8266

#### I2C

The ESP8266 does not have hardware I2C support, but can use the *i2c_soft — Software I2C* driver instead. By default two *i2c — I2C* compatability devices are instantiated:

| Device | SCL | SDA |
|---|---|---|
| i2c0 | GPIO5 | GPIO4 |
| i2c1 | GPIO12 | GPIO13 |

---

Hardware reference: https://github.com/eerimoq/hardware-reference/tree/master/esp8266

Source code: src/mcus/esp8266/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**EXTI_DEVICE_MAX**

**SPI_DEVICE_MAX**

**UART_DEVICE_MAX**

**ADC_DEVICE_MAX**

**FLASH_DEVICE_MAX**

**I2C_DEVICE_MAX**

### linux — Linux

Source code: src/mcus/linux/mcu.h

---

### Defines

**PIN_DEVICE_MAX**

**EXTI_DEVICE_MAX**

**SPI_DEVICE_MAX**

**UART_DEVICE_MAX**

**CAN_DEVICE_MAX**

**PWM_DEVICE_MAX**

**ADC_DEVICE_MAX**

**FLASH_DEVICE_MAX**

**DAC_DEVICE_MAX**

**I2C_DEVICE_MAX**

## `nrf52840` — nRF52840

Source code: src/mcus/nrf52840/mcu.h

---

### Defines

**PIN_DEVICE_MAX**

**UART_DEVICE_MAX**

**I2C_DEVICE_MAX**

**FLASH_DEVICE_MAX**

## `pic32mm0256gpm048` — PIC32MM0256GPM048

---

Source code: src/mcus/pic32mm0256gpm048/mcu.h

---

### Defines

**PIN_DEVICE_MAX**

**UART_DEVICE_MAX**

**I2C_DEVICE_MAX**

**FLASH_DEVICE_MAX**

### sam3x8e — SAM3X8E

Source code: src/mcus/sam/mcu.h

---

#### Defines

**SAM_PA**

**SAM_PB**

**SAM_PC**

**SAM_PD**

### spc56d40l1 — SPC56D40L1

Source code: src/mcus/spc56d40l1/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**UART_DEVICE_MAX**

**FLASH_DEVICE_MAX**

**CAN_DEVICE_MAX**

**I2C_DEVICE_MAX**

### stm32f100rb — STM32F100RB

Source code: src/mcus/stm32f100rb/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**UART_DEVICE_MAX**

**SPI_DEVICE_MAX**

**I2C_DEVICE_MAX**

**CAN_DEVICE_MAX**

**FLASH_DEVICE_MAX**

### stm32f205rg — STM32F205RG

Source code: src/mcus/stm32f205rg/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**UART_DEVICE_MAX**

**SPI_DEVICE_MAX**

**I2C_DEVICE_MAX**

**CAN_DEVICE_MAX**

**FLASH_DEVICE_MAX**

### stm32f303vc — STM32F303VC

Source code: src/mcus/stm32f303vc/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**UART_DEVICE_MAX**

**SPI_DEVICE_MAX**

**I2C_DEVICE_MAX**

**CAN_DEVICE_MAX**

**FLASH_DEVICE_MAX**

### xvisor_virt_v8 — Xvisor Virt-v8

Source code: src/mcus/xvisor_virt_v8/mcu.h

---

#### Defines

**PIN_DEVICE_MAX**

**UART_DEVICE_MAX**

**I2C_DEVICE_MAX**

**FLASH_DEVICE_MAX**

## 1.7 License

The Simba project as a whole is licensed under the following MIT license:

The MIT License (MIT)

Copyright (c) 2014-2018, Erik Moqvist

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following files are subjected to other licenses:

- 3pp/*: Various licenses.
- src/filesystems/fat16.*: GNU LGPL License

## 1.8 Videos

### 1.8.1 #6 Simba: CAN client-server test suite on Nano32 (ESP32) and Arduino Due.

Transmit CAN frames between a Nano32 and an Arduino Due.

### 1.8.2 #5 Simba: Room temperature (DS18B20).

Read and print the room temperature measured with a DS18B20 sensor.

### 1.8.3 #4 Simba: Hello world.

This application prints "Hello world!" to standard output.

### 1.8.4 #3 Simba: Analog read.

Read the value of an analog pin periodically once every second and print the read value to standard output.

### 1.8.5 #2 Simba: Blink example.

This video demonstrates the classic blink application. It's run on a Arduino Due that has a SAM2X8E ARM MCU.

### 1.8.6 #1 Simba: Gource of the Simba repository.

Gource visualizes the Simba Git repository file tree over time. In this project the source, test and documentation was written simultaneously, a perfect school book example of software development.

## 1.9 Links

This page contains links to external websites that are related to Simba.

Feel free to add your project to the list by submitting a pull request of this page on Github.

### 1.9.1 Pumbaa - MicroPython on Simba

Python on microcontrollers thanks to MicroPython (and in this case Simba).

Documentation: http://pumbaa.readthedocs.io

Github: https://github.com/eerimoq/pumbaa

MicroPython: http://www.micropython.org

### 1.9.2 Wingfence

A BWF for a home made robot mower.

Github: https://github.com/wingstar74/wingfence

CHAPTER 2

# Features

- *Threads* scheduled by a priority based cooperative or preemptive scheduler.
- Channels for inter-thread communication (*Queue*, *Event*).
- *Timers*.
- *Counting semaphores*.
- Device drivers (*SPI*, *UART*, . . . )
- A simple *shell*.
- *Logging*.
- Internet protocols (*TCP*, *UDP*, *HTTP*, . . . ).
- *Debug file system*.
- File systems (*FAT16*, *SPIFFS*).

See the *Library Reference* for a full list of features.

CHAPTER 3

# Testing

To ensure high code quility each module is tested extensively by many test suites. See *Testing* for details.

CHAPTER 4

# Design goals

- Rapid development.
- Clean interfaces.
- Small memory footprint.
- No dynamic memory allocation.
- Portability.

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

# Index

## Symbols



## A



## B

## C

## D

## E

## F

## G



## H

## I

## J

## L



## M

## N

## O



## P

## Q



## R



## S

## T

## U

## V



## W

## X