Contents

1 Videos ........................................ 3
2 Features ....................................... 461
3 Testing ......................................... 463
4 Design goals .................................... 465
5 Indices and tables .............................. 467
Python Module Index ............................ 469
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
Getting Started

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.

```plaintext
  → 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
  → 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.


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.

```plaintext
$ 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 avr-libc && 
  sudo apt install bossa-cli gcc-arm-none-eabi && 
  sudo apt install make unrrar autoconf automake libtool python g++ gperf 
    flex bison texinfo gawk ncurses-dev libexpat-dev 
    python-serial sed libtool-bin pmcabe 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) && 
  → 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.

```plaintext
$ 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.

**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.

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

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

    return (0);
}
```

**Makefile**

*Makefile* contains build configuration of the application.

```make
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 straightforward. 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 convenient 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
    |   |  - module2
    |  - setup.py
```

Chapter 1. Videos
Now, build and run the test suite to make sure the empty module implementation compiles and can be executed.

```bash
$ 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 if 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.

```bash
$ 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.

```c
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 unnecessary 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`.

Variables

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

Defines

`CONFIG_SYS_CONFIG_STRING`

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

`CONFIG_SYS_RESET_CAUSE`  
Read, and when needed clear, the reset cause at startup.

`CONFIG_SYS_PANIC_KICK_WATCHDOG`  
Kick the watchdog in `sys_panic()` before writing to the console.
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_FATAL_ASSERT
Enable fatal assertions, FATAL_ASSERT*().

CONFIG_PANIC_ASSERT
Enable panic assertions, PANIC_ASSERT*().

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_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_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_ESP_WIFI
Enable the esp_wifi driver.

CONFIG_EXTI
Enable the exti driver.

CONFIG_FLASH
Enable the flash driver.
CONFIG_I2C
   Enable the i2c driver.

CONFIG_I2C_SOFT
   Enable the i2c_soft driver.

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_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_SOFT
   Enable the uart_soft driver.

CONFIG_USB
   Enable the usb driver.

CONFIG_USB_DEVICE
   Enable the usb_device driver.

CONFIG_USB_HOST
   Enable the usb_host driver.

CONFIG_WATCHDOG
   Enable the watchdog 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_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_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_FS_CMD_DS18B20_LIST
Debug file system command to list all DS18B20 sensors on the bus.

CONFIG_FS_CMD_ESP_WIFI_STATUS
Debug file system command to print the Espressif WiFi status.

CONFIG_FS_CMD_FS_APPEND
Debug file system command to append to a file.
CONFIG_FS_CMD_FS_COUNTERS_LIST
Debug file system command to list all counters.

CONFIG_FS_CMD_FS_COUNTERS_RESET
Debug file system command to set all counters to zero.

CONFIG_FS_CMD_FS_FILESSTEMS_LIST
Debug file system command to list all registered file systems.

CONFIG_FS_CMD_FS_LIST
Debug file system command to list all registered file systems.

CONFIG_FS_CMD_FS_FORMAT
Debug file system command to format a file system.

CONFIG_FS_CMD_FS_PARAMETERS_LIST
Debug file system command to list all parameters.

CONFIG_FS_CMD_FS_READ
Debug file system command to read from a file.

CONFIG_FS_CMD_FS_REMOVE
Debug file system command to remove a file.

CONFIG_FS_CMD_FS_WRITE
Debug file system command to write to a file.

CONFIG_FS_CMD_I2C_READ
Debug file system command to read from a i2c bus.

CONFIG_FS_CMD_I2C_WRITE
Debug file system command to write to a i2c bus.

CONFIG_FS_CMD_LOG_LIST
Debug file system command to list all log objects.

CONFIG_FS_CMD_LOG_PRINT
Debug file system command to create a log entry and print it. Mainly used for debugging.

CONFIG_FS_CMD_LOG_SET_LOG_MASK
Debug file system command to set the log mask of a log object.

CONFIG_FS_CMD_NETWORK_INTERFACE_LIST
Debug file system command to list all network interfaces.

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

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

CONFIG_FS_CMD_PIN_WRITE
Debug file system command to write a value to a pin.

CONFIG_FS_CMD_PING_PING
Debug file system command to ping a host.

CONFIG_FS_CMD_SERVICE_LIST
Debug file system command to list all services.

CONFIG_FS_CMD_SERVICE_START
Debug file system command to start a service.
CONFIG_FS_CMD_SERVICE_STOP
  Debug file system command to stop a service.

CONFIG_FS_CMD_SETTINGS_LIST
  Debug file system command to list all settings.

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

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

CONFIG_FS_CMD_SETTINGS_WRITE
  Debug file system command to write a value to a setting.

CONFIG_FS_CMD_SYS_CONFIG
  Debug file system command to print the system configuration.

CONFIG_FS_CMD_SYS_INFO
  Debug file system command to print the system information.

CONFIG_FS_CMD_SYS_UPTIME
  Debug file system command to print the system uptime.

CONFIG_FS_CMD_SYS_PANIC
  Debug file system command to force a panic of the system.

CONFIG_FS_CMD_SYS_REBOOT
  Debug file system command to reboot the system.

CONFIG_FS_CMD_SYS_BACKTRACE
  Debug file system command to print a backtrace.

CONFIG_FS_CMD_SYS_RESET_CAUSE
  Debug file system command to print the system reset cause.

CONFIG_FS_CMD_THRD_LIST
  Debug file system command to list threads’ information.

CONFIG_FS_CMD_THRD_SET_LOG_MASK
  Debug file system command to set the log mask of a thread.

CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER
  Debug file system command to enter the application.

CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE
  Debug file system command to erase the application.

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

CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER
  Debug file system command to enter the bootloader.

CONFIG_FS_CMD_USB_DEVICE_LIST
  Debug file system command to list all USB devices.

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

CONFIG_FS_CMD_NVM_READ
  Debug file system command to read for non-volatile memory.
**CONFIG_FS_CMD_NVM_WRITE**
Debug file system command to write for non-volatile memory.

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

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

**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 baudrate.

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

CONFIG_START_FILESYSTEM
Configure a default 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_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_STD_OUTPUT_BUFFER_MAX
Maximum number of bytes in the print output buffer.

CONFIG_FLOAT
Use floating point numbers instead of integers where applicable.

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_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 columns 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_HEAP_MAX**  
Size of the harness heap, required for harness_mock_write() and harness_mock_read().

**CONFIG_HARNESS_MOCK_VERBOSE**  
Verbose mock framework.

**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_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_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.

---

**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:

```c
/* 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

The Simba build system is based on GNU Make.

**Targets**

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

**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`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMBA_ROOT</td>
<td>Path to the Simba root folder.</td>
</tr>
<tr>
<td>BOARD</td>
<td>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 <code>make BOARD=arduino_due</code>.</td>
</tr>
<tr>
<td>BAUDRATE</td>
<td>Serial port baudrate used by console and run targets.</td>
</tr>
<tr>
<td>SERIAL_PORT</td>
<td>Serial port used by console and run targets.</td>
</tr>
<tr>
<td>VERSION</td>
<td>The application version string. Usually on the form &lt;major&gt;.&lt;minor&gt;.&lt;revision&gt;.</td>
</tr>
<tr>
<td>SETTINGS_INI</td>
<td>Path to the settings file.</td>
</tr>
<tr>
<td>INC</td>
<td>Include paths.</td>
</tr>
<tr>
<td>SRC</td>
<td>Source files (.c, .asm, .rs).</td>
</tr>
<tr>
<td>CFLAGS_EXTRA</td>
<td>Extra flags passed to the compiler.</td>
</tr>
<tr>
<td>LD_FLAGS_EXTRA</td>
<td>Extra flags passed to the linker.</td>
</tr>
<tr>
<td>NASSERT</td>
<td>Build the application without assertions.</td>
</tr>
</tbody>
</table>
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...

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

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

```bash
> 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.

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

```bash
> 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.

```bash
> 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.

```bash
> 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.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>SIZE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n</td>
<td>Uart device request.</td>
</tr>
<tr>
<td>3</td>
<td>n</td>
<td>Pin device request.</td>
</tr>
<tr>
<td>5</td>
<td>n</td>
<td>Pwm device request.</td>
</tr>
<tr>
<td>7</td>
<td>n</td>
<td>Can device request.</td>
</tr>
<tr>
<td>9</td>
<td>n</td>
<td>I2c device request.</td>
</tr>
<tr>
<td>11</td>
<td>n</td>
<td>Spi device request.</td>
</tr>
</tbody>
</table>
```

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.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>SIZE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>Uart device response.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Pin device response.</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>Pwm device response.</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Can device response.</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>I2c device response.</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Spi device response.</td>
</tr>
</tbody>
</table>

Developer Guide

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

Contents:

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 frequencies in the microcontroller.

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

Only one MCU per board is supported. If there are two MCUs 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.

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.

```c
#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));
    }
```
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.

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
      | - 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
```
Simba Documentation, Release 15.0.3

Testing

To ensure high code quality 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.
- 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.
Fig. 1.1: The boards are (from left to right): *Arduino Nano, Arduino Mega, Arduino Due, STM32VLDISCOVERY, ESP-12E Development Board* and *ESP-01*

**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/chipid
• drivers/can
• drivers/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/adc
• drivers/analog_input_pin
• drivers/ds3231
• drivers/sd
• drivers/pin

**Arduino Nano**

• drivers/ds18b20
• drivers/analog_output_pin
• drivers/exti
• drivers/owi

**Arduino Pro Micro**

• kernel/sys
• kernel/thrd
• kernel/timer

**Arduino Uno**

**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/chan
- sync/event
- sync/queue
- sync/rwlock
- sync/sem
- collections/binary_tree
- collections/bits
- collections/circular_buffer
- collections/fifo
- collections/hash_map
- 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
• 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

**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/pin
- drivers/random
- filesystems/fs
- filesystems/spiffs

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/eeprom_soft

STM32F3DISCOVERY

STM32VLDISCOVERY

• kernel/sys
• kernel/thrd
WEMOS D1 mini

Releasing

Follow these steps to create a new release:

1. Write the new version in VERSION.txt. The version should have 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.

**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:
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 implement 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 interested 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`.

**Boards**

The boards supported by *Simba*.

**Arduino Due**
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 conversion*
- *analog_input_pin — Analog input pin*
- *can — Controller Area Network*
- *chipid — Chip identity*
- *dac — Digital to analog conversion*
- *ds18b20 — One-wire temperature sensor*
- *exti — External interrupts*
- *flash — Flash memory*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *mcp2515 — CAN BUS chipset*
- *owi — One-Wire Interface*
- *pin — Digital pins*
• sd — Secure Digital memory
• spi — Serial Peripheral Interface
• uart — Universal Asynchronous Receiver/Transmitter
• usb — Universal Serial Bus
• usb_host — Universal Serial Bus - Host

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>6144</td>
<td>1672</td>
</tr>
<tr>
<td>default-configuration</td>
<td>128000</td>
<td>10194</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_pwm</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>115200</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x000e0000</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWi-FiSSID</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICKSOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>384</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USBDEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USBDEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USBDEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

**Homepage**

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

**Mcu**

*sam3x8e*

**Arduino Mega**
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 conversion
- `analog_input_pin` — Analog input pin
- `analog_output_pin` — Analog output pin
- `ds18b20` — One-wire temperature sensor
- `ds3231` — RTC clock
- `exti` — External interrupts
- `i2c` — I2C
• i2c_soft — Software I2C
• mcp2515 — CAN BUS chipset
• nrf24l01 — Wireless communication
• owi — One-Wire Interface
• pin — Digital pins
• pwm — Pulse width modulation
• pwm_soft — Software pulse width modulation
• sd — Secure Digital memory
• spi — Serial Peripheral Interface
• uart — Universal Asynchronous Receiver/Transmitter
• uart_soft — Software Universal Asynchronous Receiver/Transmitter
• watchdog — Hardware watchdog

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>1734</td>
<td>279</td>
</tr>
<tr>
<td>default-configuration</td>
<td>64434</td>
<td>4005</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.2 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.2 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>“$”</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.2 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLEDEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>156</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
### 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 conversion
- **analog_input_pin** — Analog input pin
- analog_output_pin — Analog output pin
- ds18b20 — One-wire temperature sensor
- ds3231 — RTC clock
- exti — External interrupts
- i2c — I2C
- i2c_soft — Software I2C
- mcp2515 — CAN BUS chipset
- nrf24l01 — Wireless communication
- owi — One-Wire Interface
- pin — Digital pins
- pwm — Pulse width modulation
- pwm_soft — Software pulse width modulation
- sd — Secure Digital memory
- spi — Serial Peripheral Interface
- uart — Universal Asynchronous Receiver/Transmitter
- uart_soft — Software Universal Asynchronous Receiver/Transmitter
- watchdog — Hardware watchdog

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>1548</td>
<td>279</td>
</tr>
<tr>
<td>default-configuration</td>
<td>13340</td>
<td>1038</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 1.3 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRCulan_32</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FLASHDEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE_SET_MODE</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_12C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_12C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1.3 – continued from previous page
Table 1.3 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPTGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$ &quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>-1</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>156</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>256</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.3 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>1</td>
</tr>
</tbody>
</table>

Homepage

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

Mcu

atmega328p

Arduino Pro Micro

Pinout

[Diagram of 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 conversion
- `analog_input_pin` — Analog input pin
- `analog_output_pin` — Analog output pin
- `ds18b20` — One-wire temperature sensor
- `ds3231` — RTC clock
- `exti` — External interrupts
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `mcp2515` — CAN BUS chipset
- `nrf24l01` — Wireless communication
- `owi` — One-Wire Interface
- `pin` — Digital pins
- `pwm` — Pulse width modulation
- `pwm_soft` — Software pulse width modulation
- `sd` — Secure Digital memory
- `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

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>6578</td>
<td>520</td>
</tr>
<tr>
<td>default-configuration</td>
<td>15084</td>
<td>1165</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATALASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.4 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESISTEMS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSETTINGS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSETTINGS_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSETTINGS_RESET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSETTINGS_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYS_BACKTRACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYS_CONFIG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYS_INFO</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYS_PANIC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYS_REBOOT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYS_RESET_CAUSE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYS_UPTIME</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYTHRD_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMDSYTHRD_SET_LOG_MASK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMUDM_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMUDM_UPGRADEAPPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMUDM_UPGRADEAPPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMUDM_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMUSBDEVICES_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMUSBHOST_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMUPATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.4 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWNI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.4 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_USB_CDC</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIOR</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>-1</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>156</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>1</td>
</tr>
</tbody>
</table>

**Homepage**

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

**Mcu**

*atmega32u4*

**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 conversion*
- *analog_input_pin — Analog input pin*
- *analog_output_pin — Analog output pin*
- *ds18b20 — One-wire temperature sensor*
- *ds3231 — RTC clock*
- *exti — External interrupts*
- *i2c — I2C*
- *i2c_soft — Software I2C*
- *mcp2515 — CAN BUS chipset*
- *nrf24l01 — Wireless communication*
- *owi — One-Wire Interface*
- *pin — Digital pins*
- *pwm — Pulse width modulation*
- *pwm_soft — Software pulse width modulation*
- *sd — Secure Digital memory*
- *spi — Serial Peripheral Interface*
- *uart — Universal Asynchronous Receiver/Transmitter*
• **uart_soft** — *Software Universal Asynchronous Receiver/Transmitter*
• **watchdog** — *Hardware watchdog*

### 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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>1548</td>
<td>279</td>
</tr>
<tr>
<td>default-configuration</td>
<td>13340</td>
<td>1038</td>
</tr>
</tbody>
</table>

### Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.5 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FLASHDEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.5 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAEM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.5 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>-1</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>156</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>1</td>
</tr>
</tbody>
</table>

**Homepage**

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

**Mcu**

*atmega328p*
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 conversion
• analog_input_pin — Analog input pin
• ds18b20 — One-wire temperature sensor
• eeprom_soft — Software EEPROM
• esp_wifi — Espressif WiFi
• exti — External interrupts
• flash — Flash memory
• i2c — I2C
• i2c_soft — Software I2C
• led_7seg_ht16k33 — LED 7-Segment HT16K33
• owi — One-Wire Interface
• pin — Digital pins
• 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

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>265820</td>
<td>34064</td>
</tr>
<tr>
<td>default-configuration</td>
<td>312160</td>
<td>47196</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1.6 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRNT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page

1.4. Boards 75
Table 1.6 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>76800</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNECTION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x0006b000</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>0x10000</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_Prio</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_Prio</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>768</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

**Homepage**

http://espressif.com

**Mcu**

*esp8266*

**ESP-12E Development Board**
Pinout

ESP-12 pinout

- Absolute MAX per pin 12mA
- Recommended 1mA

- HIGH Run, LOW Flash

- When you use the sleep mode, IO16 and RST should be connected and IO16 will output LOW to reset the system at the time of wakeup.
- On every boot/reset/wakeup, IO15 must keep LOW, IO2 must keep HIGH.
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 baud 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 \(2^{(16 - 1)} / 8 = 4096\ \text{kB} = 4\ \text{MB}\).

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 conversion
- `analog_input_pin` — Analog input pin
- `ds18b20` — One-wire temperature sensor
- `eeprom_soft` — Software EEPROM
- `esp_wifi` — Espressif WiFi
- `exti` — External interrupts
- `flash` — Flash memory
- `i2c` — I2C
• i2c_soft — Software I2C
• led_7seg_ht16k33 — LED 7-Segment HT16K33
• owi — One-Wire Interface
• pin — Digital pins
• 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

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>265820</td>
<td>34064</td>
</tr>
<tr>
<td>default-configuration</td>
<td>312276</td>
<td>47216</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.7 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.7 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.7 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>76800</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNECTION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x00300000</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>0xFB000</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.7 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Homepage

http://espressif.com

Mcu

esp8266

ESP32-DevKitC

Pinout

![Pinout Diagram]

Default pin mapping

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

<table>
<thead>
<tr>
<th>Device function</th>
<th>GPIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED</td>
<td>16</td>
</tr>
<tr>
<td>UART 0 TX</td>
<td>1</td>
</tr>
<tr>
<td>UART 0 RX</td>
<td>3</td>
</tr>
<tr>
<td>UART 1 TX</td>
<td>19</td>
</tr>
<tr>
<td>UART 1 RX</td>
<td>23</td>
</tr>
<tr>
<td>UART 2 TX</td>
<td>18</td>
</tr>
<tr>
<td>UART 2 RX</td>
<td>5</td>
</tr>
<tr>
<td>CAN TX</td>
<td>16</td>
</tr>
<tr>
<td>CAN RX</td>
<td>17</td>
</tr>
</tbody>
</table>
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 conversion
- `analog_input_pin` — Analog input pin
- `can` — Controller Area Network
- `dac` — Digital to analog conversion
- `ds18b20` — One-wire temperature sensor
- `eeprom_soft` — Software EEPROM
- `esp_wifi` — Espressif WiFi
- `flash` — Flash memory
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `owi` — One-Wire Interface
- `pin` — Digital pins
- `random` — Random numbers.
- `spi` — Serial Peripheral Interface
- `uart` — Universal Asynchronous Receiver/Transmitter
- `ws2812` — NeoPixels

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.
<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>91428</td>
<td>8644</td>
</tr>
<tr>
<td>default-configuration</td>
<td>356256</td>
<td>83420</td>
</tr>
</tbody>
</table>

**Default configuration**

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSUME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ISVALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1.8 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIGSOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>115200</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNECTION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x00300000</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>0x200000</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_Prio</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_Prio</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.8 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SYSTEM_TICKSOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>8192</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>1024</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Homepage


Mcu

*esp32*

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 conversion
- *analog_input_pin* — Analog input pin
- *ds18b20* — One-wire temperature sensor
- *eeprom_soft* — Software EEPROM
- *esp_wifi* — Espressif WiFi
- *exti* — External interrupts
- *flash* — Flash memory
- *i2c* — I2C
- *i2c_soft* — Software I2C
- *led_7seg_ht16k33* — LED 7-Segment HT16K33
- *owi* — One-Wire Interface
- *pin* — Digital pins
- *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*

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>265820</td>
<td>34064</td>
</tr>
<tr>
<td>default-configuration</td>
<td>311452</td>
<td>46672</td>
</tr>
</tbody>
</table>

**Default configuration**

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_Start</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OW1</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT.Pin</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.9 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASHDEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>76800</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.9 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START.Console_USB_CDC_WAIT_FOR_CONNECTION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Homepage

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

esp8266

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 conversion
- `analog_input_pin` — Analog input pin
- `analog_output_pin` — Analog output pin
- `can` — Controller Area Network
• *dac* — Digital to analog conversion
• *ds18b20* — One-wire temperature sensor
• *eeprom_soft* — Software EEPROM
• *exti* — External interrupts
• *flash* — Flash memory
• *i2c* — I2C
• *i2c_soft* — Software I2C
• *owi* — One-Wire Interface
• *pin* — Digital pins
• *pwm* — Pulse width modulation
• *pwm_soft* — Software pulse width modulation
• *random* — Random numbers.
• *sd* — Secure Digital memory
• *spi* — Serial Peripheral Interface
• *uart* — Universal Asynchronous Receiver/Transmitter

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>145132</td>
<td>320056</td>
</tr>
<tr>
<td>default-configuration</td>
<td>385403</td>
<td>474608</td>
</tr>
</tbody>
</table>

**Default configuration**

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.10 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>1028</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.10 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>1024</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>1024</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.10 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Homepage

http://www.kernel.org

Mcu

linux

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 conversion
• analog_input_pin — Analog input pin
• can — Controller Area Network
• dac — Digital to analog conversion
• ds18b20 — One-wire temperature sensor
• eeprom_soft — Software EEPROM
• esp_wifi — Espressif WiFi
• flash — Flash memory
• i2c — I2C
• i2c_soft — Software I2C
• owi — One-Wire Interface
• pin — Digital pins
• random — Random numbers.
• spi — Serial Peripheral Interface
• uart — Universal Asynchronous Receiver/Transmitter
• ws2812 — NeoPixels

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>91428</td>
<td>8644</td>
</tr>
<tr>
<td>default-configuration</td>
<td>356256</td>
<td>83420</td>
</tr>
</tbody>
</table>
Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td></td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FSFILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1.11 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FSTORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OW1</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OW1</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.11 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMITTED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>115200</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x00300000</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>0x20000</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.11 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>8192</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>1024</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USBDEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USBDEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USBDEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Homepage

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

Mcu

*esp32*

Nano32

Pinout

![Nano32 ESP32 Pinout](image)

Default pin mapping

Here is a list of additional pin mappings not part of the picture above.
<table>
<thead>
<tr>
<th>Device function</th>
<th>GPIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART 0 TX</td>
<td>1</td>
</tr>
<tr>
<td>UART 0 RX</td>
<td>3</td>
</tr>
<tr>
<td>UART 1 TX</td>
<td>19</td>
</tr>
<tr>
<td>UART 1 RX</td>
<td>23</td>
</tr>
<tr>
<td>UART 2 TX</td>
<td>18</td>
</tr>
<tr>
<td>UART 2 RX</td>
<td>5</td>
</tr>
<tr>
<td>CAN TX</td>
<td>16</td>
</tr>
<tr>
<td>CAN RX</td>
<td>17</td>
</tr>
</tbody>
</table>

**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 conversion
- `analog_input_pin` — Analog input pin
- `can` — Controller Area Network
- `dac` — Digital to analog conversion
- `ds18b20` — One-wire temperature sensor
- `eeprom_soft` — Software EEPROM
- `esp_wifi` — Espressif WiFi
- `flash` — Flash memory
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `owi` — One-Wire Interface
- `pin` — Digital pins
- `random` — Random numbers.
- `spi` — Serial Peripheral Interface
- `uart` — Universal Asynchronous Receiver/Transmitter
- `ws2812` — NeoPixels

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>91428</td>
<td>8644</td>
</tr>
<tr>
<td>default-configuration</td>
<td>356248</td>
<td>83420</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>160000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SON</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OW1</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.12 – continued from previous page
Table 1.12 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>115200</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNECTION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x00300000</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>0x20000</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.12 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>8192</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>1024</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Homepage

http://esp32.de

Mcu

esp32

NodeMCU
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 conversion
- `analog_input_pin` — Analog input pin
- `ds18b20` — One-wire temperature sensor
- `eeprom_soft` — Software EEPROM
- `esp_wifi` — Espressif WiFi
- `exti` — External interrupts
- `flash` — Flash memory
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `led_7seg_ht16k33` — LED 7-Segment HT16K33
- `owi` — One-Wire Interface
- `pin` — Digital pins
- `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

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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>265820</td>
<td>34064</td>
</tr>
<tr>
<td>default-configuration</td>
<td>312316</td>
<td>47196</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.13 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_CHANGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1.13 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1.13 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>76800</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
</tbody>
</table>

*Continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNECTION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x00300000</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>0xFB000</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>
Simba Documentation, Release 15.0.3

Homepage

http://www.nodemcu.com

Mcu

esp8266

Particle IO Photon

Pinout

20 mA max per I/O pin
PWM outputs are 8-bit resolution with default frequency of 500 Hz.
VIN (input) = 3.6 V to 5.5 V
VIN (output) = approx. 4.8 V when powered from USB
Avg. current with Wi-Fi on = 100 mA

Detailed pinout
### Right side pins

<table>
<thead>
<tr>
<th>USB</th>
<th>Pin</th>
<th>Exposed Functions</th>
<th>STM32 Pin</th>
<th>PØ Pin #</th>
<th>PØ Pin Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3V3</td>
<td>3V3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>RST</td>
<td></td>
<td>E8</td>
<td>26</td>
<td>MICRO_RST_N</td>
</tr>
<tr>
<td>VBAT</td>
<td>VBAT</td>
<td></td>
<td>A9</td>
<td>28</td>
<td>VBAT</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>JTAG_TMS</td>
<td></td>
<td>PA13</td>
<td>44</td>
<td>MICRO_JTAG_TMS</td>
</tr>
<tr>
<td>D6</td>
<td>JTAG_TCK</td>
<td></td>
<td>PA14</td>
<td>40</td>
<td>MICRO_JTAG_TCK</td>
</tr>
<tr>
<td>D5</td>
<td>JTAG_TDI</td>
<td>SPI3_SS</td>
<td>PA15</td>
<td>43</td>
<td>MICRO_JTAG_TDI</td>
</tr>
<tr>
<td>D4</td>
<td>JTAG_TDO</td>
<td>SPI3_SCK</td>
<td>PB3</td>
<td>41</td>
<td>MICRO_JTAG_TDO</td>
</tr>
<tr>
<td>D3</td>
<td>JTAG_TRST</td>
<td>SPI3_MISO</td>
<td>PB4</td>
<td>42</td>
<td>MICRO_JTAG_TRSTN</td>
</tr>
<tr>
<td>D2</td>
<td>SPI3_MOSI</td>
<td>CAN2_RX</td>
<td>PB5</td>
<td>3</td>
<td>MICRO_GPIO_5</td>
</tr>
<tr>
<td>D1</td>
<td>SCL</td>
<td>CAN2_TX</td>
<td>PB6</td>
<td>5</td>
<td>MICRO_GPIO_3</td>
</tr>
<tr>
<td>D0</td>
<td>SDA</td>
<td>TIM4_CH2</td>
<td>PB7</td>
<td>4</td>
<td>MICRO_GPIO_4</td>
</tr>
</tbody>
</table>

### Left side pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>USB</th>
<th>Exposed Functions</th>
<th>STM32 Pin</th>
<th>PØ Pin #</th>
<th>PØ Pin Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIN</td>
<td>VIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td></td>
<td>USART1_TX TIM1.CH2</td>
<td>PA9</td>
<td>39</td>
<td>MICRO_UART_TX</td>
</tr>
<tr>
<td>RX</td>
<td></td>
<td>USART1_RX TIM1.CH3</td>
<td>PA10</td>
<td>38</td>
<td>MICRO_UART_RX</td>
</tr>
<tr>
<td>WKP</td>
<td></td>
<td></td>
<td>PA0</td>
<td>27</td>
<td>MICRO_WKUP</td>
</tr>
<tr>
<td>ADC</td>
<td></td>
<td>ADC0 TIM5.CH1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td></td>
<td>SPI1.MOSI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td></td>
<td>SPI1.MISO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td></td>
<td>SPI1.SCK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td>ADC12 SPI1_SS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td></td>
<td>ADC13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.4. Boards
User I/O

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

Prerequisites

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/udev/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.

- `eeprom_soft` — Software EEPROM
- `flash` — Flash memory
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `pin` — Digital pins
- `uart` — Universal Asynchronous Receiver/Transmitter

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>2668</td>
<td>1672</td>
</tr>
<tr>
<td>default-configuration</td>
<td>63848</td>
<td>5998</td>
</tr>
</tbody>
</table>
Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.14 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1.14 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>384</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

**Homepage**

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

**Mcu**

*stm32f205rg*

**SPC56D Discovery**

**Pinout**
**Pin functions**

These are the default pin functions in Simba.

<table>
<thead>
<tr>
<th>List</th>
<th>Index</th>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>12</td>
<td>PB0</td>
<td>CAN TX</td>
</tr>
<tr>
<td>X2</td>
<td>9</td>
<td>PB1</td>
<td>CAN RX</td>
</tr>
<tr>
<td>X1</td>
<td>10</td>
<td>PB2</td>
<td>UART0 TX</td>
</tr>
<tr>
<td>X2</td>
<td>10</td>
<td>PB3</td>
<td>UART0 RX</td>
</tr>
<tr>
<td>X4</td>
<td>6</td>
<td>PC2</td>
<td>LED (D8)</td>
</tr>
</tbody>
</table>

**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.

- `can` — Controller Area Network
- `eeprom_soft` — Software EEPROM
- `flash` — Flash memory
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `pin` — Digital pins
- `uart` — Universal Asynchronous Receiver/Transmitter
- `watchdog` — Hardware watchdog

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>13636</td>
<td>952</td>
</tr>
<tr>
<td>default-configuration</td>
<td>75308</td>
<td>5954</td>
</tr>
</tbody>
</table>

## Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.15 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_FS_RM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page.
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>1028</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>115200</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_Prio</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_Prio</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>156</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>1</td>
</tr>
</tbody>
</table>

**Homepage**


**Mcu**

*spc56d40l1*

**STM32F3DISCOVERY**
Pinout

These are the default pin functions in Simba.
<table>
<thead>
<tr>
<th>Function</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART0 TX</td>
<td>PA9</td>
</tr>
<tr>
<td>UART0 RX</td>
<td>PA10</td>
</tr>
<tr>
<td>UART1 TX</td>
<td>PA2</td>
</tr>
<tr>
<td>UART1 RX</td>
<td>PA3</td>
</tr>
<tr>
<td>UART2 TX</td>
<td>PB10</td>
</tr>
<tr>
<td>UART2 RX</td>
<td>PB11</td>
</tr>
<tr>
<td>SPI0 SCK</td>
<td>PA5</td>
</tr>
<tr>
<td>SPI0 MISO</td>
<td>PA6</td>
</tr>
<tr>
<td>SPI0 MOSI</td>
<td>PA7</td>
</tr>
<tr>
<td>SPI1 SCK</td>
<td>PA13</td>
</tr>
<tr>
<td>SPI1 MISO</td>
<td>PA14</td>
</tr>
<tr>
<td>SPI1 MOSI</td>
<td>PA15</td>
</tr>
<tr>
<td>SPI2 SCK</td>
<td>PC10</td>
</tr>
<tr>
<td>SPI2 MISO</td>
<td>PC11</td>
</tr>
<tr>
<td>SPI2 MOSI</td>
<td>PC12</td>
</tr>
<tr>
<td>I2C0 SCL</td>
<td>PB8</td>
</tr>
<tr>
<td>I2C0 SDA</td>
<td>PB9</td>
</tr>
<tr>
<td>I2C1 SCL</td>
<td>PF0</td>
</tr>
<tr>
<td>I2C1 SDA</td>
<td>PF1</td>
</tr>
<tr>
<td>CAN TX</td>
<td>PD1</td>
</tr>
<tr>
<td>CAN RX</td>
<td>PD0</td>
</tr>
</tbody>
</table>

**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.

- `eeprom_soft` — Software EEPROM
- `flash` — Flash memory
- `i2c` — I²C
- `i2c_soft` — Software I²C
- `pin` — Digital pins
- `uart` — Universal Asynchronous Receiver/Transmitter

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>2816</td>
<td>1672</td>
</tr>
<tr>
<td>default-configuration</td>
<td>62976</td>
<td>5502</td>
</tr>
</tbody>
</table>

Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_LOG_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.16 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASHDEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>384</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

**Homepage**


**Mcu**

stm32f303vc

**STM32VLDISCOVERY**
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
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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART0 TX</td>
<td>PA9</td>
</tr>
<tr>
<td>UART0 RX</td>
<td>PA10</td>
</tr>
<tr>
<td>UART1 TX</td>
<td>PA2</td>
</tr>
<tr>
<td>UART1 RX</td>
<td>PA3</td>
</tr>
<tr>
<td>UART2 TX</td>
<td>PC10</td>
</tr>
<tr>
<td>UART2 RX</td>
<td>PC11</td>
</tr>
<tr>
<td>SPI0 SCK</td>
<td>PA5</td>
</tr>
<tr>
<td>SPI0 MISO</td>
<td>PA6</td>
</tr>
<tr>
<td>SPI0 MOSI</td>
<td>PA7</td>
</tr>
<tr>
<td>I2C0 SCL</td>
<td>PB8</td>
</tr>
<tr>
<td>I2C0 SDA</td>
<td>PB9</td>
</tr>
</tbody>
</table>

**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.

- `eeprom_soft` — Software EEPROM
- `flash` — Flash memory
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `pin` — Digital pins
- `uart` — Universal Asynchronous Receiver/Transmitter

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>2816</td>
<td>1672</td>
</tr>
<tr>
<td>default-configuration</td>
<td>64256</td>
<td>6006</td>
</tr>
</tbody>
</table>

### Default configuration

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESSMocks_VERBOSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.17 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.17 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>&quot;$&quot;</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIGSOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAMCLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>38400</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOL_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSO_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSO_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSO_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>65536</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
Table 1.17 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>840</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>384</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

**Homepage**


**Mcu**

`stm32f100rb`

**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 conversion
- `analog_input_pin` — Analog input pin
- `ds18b20` — One-wire temperature sensor
- `eeprom_soft` — Software EEPROM
- `esp_wifi` — Espressif WiFi
- `exti` — External interrupts
- `flash` — Flash memory
- `i2c` — I2C
- `i2c_soft` — Software I2C
- `led_7seg_ht16k33` — LED 7-Segment HT16K33
- `owi` — One-Wire Interface
- `pin` — Digital pins
- `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*

**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.

<table>
<thead>
<tr>
<th>Application</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimal-configuration</td>
<td>265820</td>
<td>34064</td>
</tr>
<tr>
<td>default-configuration</td>
<td>311488</td>
<td>46700</td>
</tr>
</tbody>
</table>

**Default configuration**

Default Standard Library configuration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_ASSERT_FORCE_FATAL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CAN_FRAME_TIMESTAMP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_CRC_TABLE_LOOKUP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DEBUG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC</td>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_32</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_CRC_CCITT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EEPROM_SOFT_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EMACS_COLUMNS_MAX</td>
<td>80</td>
</tr>
<tr>
<td>CONFIG_EMACS_HEAP_SIZE</td>
<td>32768</td>
</tr>
<tr>
<td>CONFIG_EMACS_ROWS_MAX</td>
<td>24</td>
</tr>
<tr>
<td>CONFIG_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_EXTERNAL_OSCILLATOR_FREQUENCY_HZ</td>
<td>16000000</td>
</tr>
<tr>
<td>CONFIG_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FAT16</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FATAL_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FILESYSTEM_GENERIC</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLASH_DEVICE_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FLOAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_DS18B20_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_ESP_WIFI_STATUS</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_APPEND</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_COUNTERS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FILESYSTEMS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_FORMAT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_PARAMETERS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_REMOVE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_FS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_I2C_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_PRINT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_LOG_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NETWORK_INTERFACE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_SET_MODE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_SET_LOG_MASK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_ERASE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_APPLICATION_IS_VALID</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_UPGRADE_BOOTLOADER_ENTER</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 1.18 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_PATH_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE</td>
<td>512</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX</td>
<td>2048</td>
</tr>
<tr>
<td>CONFIG_HARNESS_LOG_HIGH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS</td>
<td>300</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_EXPECT_BUFFER_SIZE</td>
<td>128</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ADC</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS3231</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SSL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_STD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 1.18 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_MODULE_INIT_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US</td>
<td>2000000</td>
</tr>
<tr>
<td>CONFIG_NRF24L01</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_0_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_BLOCK_1_SIZE</td>
<td>16384</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_CHUNK_SIZE</td>
<td>(CONFIG_NVM_SIZE + 8)</td>
</tr>
<tr>
<td>CONFIG_NVM_EEPROM_SOFT_FLASH_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_NVM_SIZE</td>
<td>2040</td>
</tr>
<tr>
<td>CONFIG_OWI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PANIC_ASSERT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PIN</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PREEMPTIVE_SCHEDULER</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PROFILE_STACK</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_PWM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_PWM_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_RANDOM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SD</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_AREA_SIZE</td>
<td>256</td>
</tr>
<tr>
<td>CONFIG_SETTINGS_BLOB</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SHELL_COMMAND_MAX</td>
<td>64</td>
</tr>
<tr>
<td>CONFIG_SHELL_HISTORY_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_SHELL_MINIMAL</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SHELL_PROMPT</td>
<td>“$”</td>
</tr>
<tr>
<td>CONFIG_SHT3XD</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOAM_EMBEDDED_DATABASE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SOCKET_RAW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SOFTWARE_I2C</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_BOOT_ENTRY_RCHW</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPC5_RAM_CLEAR_ALL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPI</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SPIFFS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE</td>
<td>CONFIG_START_CONSOLE_UART</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_DEVICE_INDEX</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_BAUDRATE</td>
<td>76800</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_UART_RX_BUFFER_SIZE</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_CONTROL_INTERFACE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_IN</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_ENDPOINT_OUT</td>
<td>3</td>
</tr>
<tr>
<td>CONFIG_START_CONSOLE_USB_CDC_WAIT_FOR_CONNETION</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_FILESYSTEM_ADDRESS</td>
<td>0x00300000</td>
</tr>
</tbody>
</table>

Continued on next page

1.4. Boards
Table 1.18 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_START_FILESYSTEM_SIZE</td>
<td>0xFB000</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_CONNECT_TIMEOUT</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD</td>
<td>MyWiFiPassword</td>
</tr>
<tr>
<td>CONFIG_START_NETWORK_INTERFACE_WIFI_SSID</td>
<td>MyWiFiSSID</td>
</tr>
<tr>
<td>CONFIG_START_NVM</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SHELL_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_START_SOAM</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_PRIO</td>
<td>30</td>
</tr>
<tr>
<td>CONFIG_START_SOAM_STACK_SIZE</td>
<td>1536</td>
</tr>
<tr>
<td>CONFIG_STD_OUTPUT_BUFFER_MAX</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPTS</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_INTERRUPT_STACK_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_FREQUENCY</td>
<td>100</td>
</tr>
<tr>
<td>CONFIG_SYSTEM_TICK_SOFTWARE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_CONFIG_STRING</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_SYS_PANIC_KICK_WATCHDOG</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_SYS_RESET_CAUSE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_SYS_SIMBA_MAIN_STACK_MAX</td>
<td>4096</td>
</tr>
<tr>
<td>CONFIG_THRD_CPU_USAGE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_DEFAULT_LOG_MASK</td>
<td>LOG_UPTO(INFO)</td>
</tr>
<tr>
<td>CONFIG_THRD_ENV</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_IDLE_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_MONITOR_STACK_SIZE</td>
<td>768</td>
</tr>
<tr>
<td>CONFIG_THRD_SCHEDULED</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_STACK_HEAP_SIZE</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_THRD_TERMINATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_TIME_UNIX_TIME_TO_DATE</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_UART_SOFT</td>
<td>1</td>
</tr>
<tr>
<td>CONFIG_USB</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_PID</td>
<td>0x8037</td>
</tr>
<tr>
<td>CONFIG_USB_DEVICE_VID</td>
<td>0x2341</td>
</tr>
<tr>
<td>CONFIG_USB_HOST</td>
<td>0</td>
</tr>
<tr>
<td>CONFIG_WATCHDOG</td>
<td>0</td>
</tr>
</tbody>
</table>

Homepage

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

Mcu

esp8266
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.

Analog Read

About

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

Source code

```c
/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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. */
```
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

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.

Source code

```c
/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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:
 * ```
#include "simba.h"

```c
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
```

## Blink
About

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

Source code

```c
/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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. */
        pin_toggle(&led);
    }
}
```
The source code can also be found on Github in the examples/blink folder.

**Build and run**

Build and upload the application.

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

**DS18B20**

**About**

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

**Source code**

```c
/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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()
```

1.5. Examples
```c
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();
owni_init(&owi, &pin_d7_dev, devices, membersof(devices));
usi2b20_init(&ds, &owi);
time_busy_wait_us(50000);

/* 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

Filesystem

About

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

Source code

/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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

1.5. Examples
/**
 * 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'.\n"));

    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\n"), counter);

    return (0);
}
```c
int main() {
    int res;
    sys_start();
    std_printf(sys_get_info());

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

    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.
```

**Hello World**

**About**

This application prints “Hello world!” to standard output.

**Source code**

```c
/**
 * @section License
```
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!
$  
```

**HTTP Client**
About

Connect to a remote host perform a HTTP GET action to fetch the root page ‘/’ from the remote host.

Define `CONFIG_START_NETWORK_INTERFACE_WIFI_SSID` and `CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD` in `config.h` to the SSID and password of your WiFi, otherwise the board will fail to connect to the WiFi network. Alternatively, the defines can be given as defines on the make command line as seen in the example below.

Source code

```c
/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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'.\n"), remote_host_ip);

if (inet_aton(remote_host_ip, &remote_host_address.ip) != 0) {
    std_printf(FSTR("Bad ip address '.'.\n"), remote_host_ip);
    return (-1);
}

remote_host_address.port = 80;

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

std_printf(FSTR("First 64 bytes of the response:\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. It must be built for ESP12E or ESP01 since those are the only boards with a network connection (WiFi).

```
$ cd examples/http_client
$ make -s BOARD=esp12e CDEFS_EXTRA="CONFIG_START_NETWORK_INTERFACE_WIFI SSID=Qvist, CONFIG_START_NETWORK_INTERFACE_WIFI_PASSWORD=FooBar" 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
...
$

Ping

About

Ping a remote host periodically once every second.

Source code

```c
/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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 216.58.211.142

int main()
```
int res, attempt;
char remote_host_ip[] = STRINGIFY(REMOTE_HOST_IP);
struct inet_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);
}

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.

```bash
$ cd examples/ping
$ make -s BOARD=<board> run
Successfully pinged '192.168.1.100' in 20 ms (#1).
Successfully pinged '192.168.1.100' in 20 ms (#2).
Successfully pinged '192.168.1.100' in 20 ms (#3).
```

**Queue**
About

Use a queue to communicate between two threads.

Source code

```c
/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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\n"), value);

return (0);

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

Shell

About

Use the serial port to monitor and control the application.

Source code

/**
 * @section License
 *
 * The MIT License (MIT)
 *
 * Copyright (c) 2014-2017, 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
The source code can also be found on Github in the examples/shell folder.

1.5. Examples
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

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATE</th>
<th>PRIO</th>
<th>CPU</th>
<th>MAX-STACK-USAGE</th>
<th>LOGMASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>shell</td>
<td>current</td>
<td>0</td>
<td>0%</td>
<td>358/5575</td>
<td>0x0f</td>
</tr>
<tr>
<td>idle</td>
<td>ready</td>
<td>127</td>
<td>0%</td>
<td>57/156</td>
<td>0x0f</td>
</tr>
</tbody>
</table>
```

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.

Source code

```c
/**
 * @section License
 * *
 * The MIT License (MIT)
 * *
 * Copyright (c) 2014-2017, 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
```
The source code can also be found on Github in the examples/timer folder.
Build and run

Build and upload the application.

```shell
$ cd examples/timer
$ make -s BOARD=<board> run
timeout
timeout
timeout
```

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 implementations are available on Github.

kernel

The kernel package is the heart in Simba. It implements the thread scheduler.

The kernel package on Github.

assert — Assertions

Source code: src/kernel/assert.h

---

Defines

**FATAL** (n)

**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

Source code: [src/kernel/errno.h](#)
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.
ELNRRNG
   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.

EBADRC
   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.
ENOBIFS
    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.
**sys — System**

System level functionality and definitions.

---

**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) __attribute__((noreturn))`

**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)`
  Conversion from the time struct to system ticks.
- `static void st2t (sys_tick_t tick, struct time_t *time_p)`
  Conversion 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** (const char *message_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**

• message_p: Panic message to write to the console.

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.

do 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.

do sys_set_stdin (void *chan_p)
Set the standard input channel.

    Return void.

    Parameters
        • chan_p: New standard input channel.

do *sys_get_stdin (void)
Get the standard input channel.

    Return Standard input channel.

do sys_set_stdout (void *chan_p)
Set the standard output channel.

    Return void.

    Parameters
        • chan_p: New standard output channel.

do *sys_get_stdout (void)
Get the standard output channel.

    Return Standard output channel.

do 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.

Variables

const char *sys_reset_cause_string_map[sys_reset_cause_max_t]
  System reset cause strings map.

struct sys_t sys

struct sys_t
Public Members

sys_on_fatal_fn_t on_fatal_callback
void *stdin_p
void *stdout_p
uint32_t start
uint32_t time
struct sys_t::interrupt

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 running 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Print a list of all threads.</td>
</tr>
<tr>
<td>set_log_mask &lt;thread name&gt; &lt;mask&gt;</td>
<td>Set the log mask of thread &lt;thread name&gt; to mask.</td>
</tr>
<tr>
<td>monitor/set_period_ms &lt;ms&gt;</td>
<td>Set the monitor thread sampling period to &lt;ms&gt; milliseconds.</td>
</tr>
<tr>
<td>monitor/set_print &lt;state&gt;</td>
<td>Enable(1)/disable(0) monitor statistics to be printed periodically.</td>
</tr>
</tbody>
</table>

#### Example output from the shell:

```
$ kernel/thrd/list

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATE</th>
<th>PRIO</th>
<th>CPU</th>
<th>SCHEDULED</th>
<th>LOGMASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>current</td>
<td>0</td>
<td>0%</td>
<td>1</td>
<td>0x0f</td>
</tr>
<tr>
<td>ready</td>
<td>127</td>
<td>0%</td>
<td>0x0f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ready</td>
<td>-80</td>
<td>0%</td>
<td>0x0f</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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)

**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**

Reschedule 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 infinite 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.

```c
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), -ETIMEOUT on timeout or other negative error code.

**Parameters**

- **timeout_p**: Time to wait to be resumed before a timeout occurs and the function returns.

```c
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().

```c
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.

```c
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.

```c
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 ms)
    Pauses the current thread for given number of milliseconds.

    Return  zero(0) or negative error code.

    Parameters
    • ms: Milliseconds to sleep.

int thrd_sleep_us (long us)
    Pauses the current thread for given number of microseconds.

    Return  zero(0) or negative error code.

    Parameters
    • us: 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  Thread 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 thread.
    • 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), -ETIMEOUT 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 threads’ 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 threads’ top of stack, or NULL on error.

int thrd_prio_list_init (struct thrd_prio_list_t *self_p)
Initialize given priority 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 Popped 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

```c
struct thrd_environment_variable_t *variables_p
size_t number_of_variables
size_t max_number_of_variables
```

Public Members

```c
struct thrd_prio_list_elem_t elem
struct thrd_t::scheduler
struct thrd_port_t port
int prio
int state
int err
int log_mask
struct timer_t *timer_p
const char *name_p
struct thrd_t *next_p
struct thrd_t::statistics
size_t stack_size
```

time — System time

Source code: src/kernel/time.h, src/kernel/time.c
Test code: tst/kernel/time/main.c
Test coverage: src/kernel/time.c

Functions

```c
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.
```

```c
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.

```c
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.

```c
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 subtracting `left_p` from `right_p`.
- `left_p`: The operand to subtract from.
- `right_p`: The operand to subtract.

```c
void time_busy_wait_us (long useconds)
```

Busy wait for given number of microseconds.

NOTE: The maximum allowed time to sleep is target specific.

**Return** void

**Parameters**
- `useconds`: Microseconds to busy wait.

```c
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.

**Parameters**
- `date_p`: Converted time.
- `time_p`: Unix time to convert.

```c
struct time_t
    #include <time.h>
```
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.

An application requiring timers with higher precision than the system tick must use the hardware timers.

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
**Typedefs**

```c
typedef void (*timer_callback_t)(void *arg_p)
Time callback prototype.
```

**Functions**

```c
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.
```

```c
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.

**Return** zero(0) or negative error code.

**Parameters**

- **self_p**: Timer object to initialize with given parameters.
- **timeout_p**: The timer timeout value.
- **callback**: Function 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.
```

```c
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.
```

```c
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()).
```

```c
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.

**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
    sys_tick_t delta
    sys_tick_t timeout
    int flags
    timer_callback_t callback
    void *arg_p

types — Common types

Source code: src/kernel/types.h

---

Defines

UNUSED (v)

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));
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

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

**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; listener 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 listener 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.

```
+--------------+ +--------------+
| listener 0   | | listener 2   |
| id:7, chan:0 | | id:9, chan:2 |
+--------------+ +--------------+
BUS =============+==============+=======
|               |
| +-------------+  
| | listener 1  |
| | id:7, chan:1 |
| +--------------+
```

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)

Detach 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.
Public Members

```c
struct rwlock_t rwlock
struct binary_tree_t listeners
struct bus_listener_t
```

Public Members

```c
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.

```
+------------+ +------------+
| thread 0 | channel 0 | thread 1 |
| +----------+ +----------+
| producer | | consumer |
+------------+ +------------+
```

In the second example, isr 0 and thread 2 communicates over a channel. isr 0 writes data to the channel and thread 2 read the written data.

```
+------------+ +------------+
| isr 0 | channel 1 | thread 2 |
| +----------+ +----------+
| producer | | consumer |
+------------+ +------------+
```

---

Source code: `src/sync/chan.h`, `src/sync/chan.c`

Test coverage: `src/sync/chan.c`

---

**Defines**

```c
CHAN_CONTROL_LOG_BEGIN
CHAN_CONTROL_LOG_END
End of a log entry.
CHAN_CONTROL_PRINTF_BEGIN
Beginning of printf output.
```
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 read from.
• 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_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.
- **filter**: 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**

1.6. Library Reference 207
• 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.

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.

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.

```c
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.

```c
int chan_list_init(struct chan_list_t *list_p, void *workspace_p, size_t size)
```

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

- **list_p**: List to initialize.
- **workspace_p**: Workspace for internal use.
- **size**: Size of the workspace in bytes.

```c
int chan_list_destroy(struct chan_list_t *list_p)
```

Destroy an initialized list of channels.

Return  zero(0) or negative error code.

Parameters

- **list_p**: List to destroy.

```c
int chan_list_add(struct chan_list_t *list_p, void *chan_p)
```

Add given channel to list of channels.

Return  zero(0) or negative error code.

Parameters

- **list_p**: List of channels.
- **chan_p**: Channel to add.

```c
int chan_list_remove(struct chan_list_t *list_p, void *chan_p)
```

Remove given channel from list of channels.

Return  zero(0) or negative error code.

Parameters

- **list_p**: List of channels.
• chan_p: Channel to remove.

```c
void *chan_list_poll (const struct chan_list_t *list_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**

• list_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.

```c
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.

```c
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.

```c
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.

```c
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.

```c
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).

```c
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).

```c
struct chan_list_t
```
Public Members

```c
struct chan_t **chans_pp
size_t max
size_t len
int flags
```

```c
struct chan_t
#include <chan.h> Channel datastructure.
```

Public Members

```c
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
```

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.

Source code: src/sync/event.h, src/sync/event.c
Test code: tst/sync/event/main.c
Test coverage: src/sync/event.c

Functions

```c
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 an event 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.

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 occurred.
•  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 is active, otherwise zero(0).

Parameters
•  self_p: Event channel object.

struct event_t

#include <event.h>

Public Members

struct chan_t base

uint32_t mask
uint32_t reader_mask
**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.

![Queue Diagram](image)

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.

---

**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**

```c
QUEUE_INIT_DECL (_name, _buf, _size)
```

** Enums**

```c
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**

```c
int queue_init (struct queue_t *self_p, void *buf_p, size_t size)
```

Initialize given queue.

**Return** zero(0) or negative error code

**Parameters**

- **self_p**: Queue to initialize.
- **buf_p**: Buffer.
- **size**: Size of buffer.

```c
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.

```c
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.

```c
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()`).

```c
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 read bytes or negative error code.

**Parameters**

- **self_p**: Queue to read from.
- **buf_p**: Buffer to read to.
- **size**: Size to read.

```c
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 written bytes or negative error code.
Parameters

- `self_p`: Queue to write to.
- `buf_p`: Buffer to write from.
- `size`: Number of bytes to write.

```c
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 written bytes or negative error code.

Parameters

- `self_p`: Queue to write to.
- `buf_p`: Buffer to write from.
- `size`: Number of bytes to write.

```c
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 queue.

Parameters

- `self_p`: Queue.

```c
ssize_t queue_unused_size( struct queue_t *self_p )
```

Get the number of unused bytes in the queue.

Return  Number of bytes unused in the queue.

Parameters

- `self_p`: Queue.

```c
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 bytes unused in the queue.

Parameters

- `self_p`: Queue.

```c
struct queue_buffer_t
```

Public Members

- `char *begin_p`
- `char *read_p`
- `char *write_p`
- `char *end_p`
size_t size
struct queue_writer_elem_t

Public Members

struct thrd_prio_list_elem_t base
void *buf_p
size_t size
size_t left

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::@103 queue_t::reader
struct queue_buffer_t buffer
queue_state_t state

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.

```c
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.

```c
struct rwlock_t
    #include <rwlock.h>
```

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 occurred. That is, `sem_give()` may be called multiple times and the semaphore resource count will remain at zero(0) until `sem_take()` is called.

Source code: src/sync/sem.h, src/sync/sem.c
Test code: tst/sync/sem/main.c
Test coverage: src/sync/sem.c

**Defines**

```c
SEM_INIT_DECL (name, _count, _count_max)
```
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()** is 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.

drivers

The drivers package on Github.

Modules:

adc — Analog to digital conversion

Source code: src/drivers/adc.h, src/drivers/adc.c
Test code: tst/drivers/adc/main.c

Defines

ADC_REFERENCE_VCC

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 architecture. The sampling rate is not used in single sample conversions, ie. calls to `adc_async_convert()` and `adc_convert()` with length one; or calls to `adc_convert_isr()`.

```c
int adc_async_convert (struct adc_driver_t *self_p, uint16_t *samples_p, size_t length)
```

Start an asynchronous conversion of analog signal to digital samples. Call `adc_async_wait()` to wait for the conversion to complete.

**Return** zero(0) or negative error code.

**Parameters**

- self_p: Driver object.
- samples_p: Converted samples.
- length: Length of samples array.

```c
int adc_async_wait (struct adc_driver_t *self_p)
```

Wait for an asynchronous conversion to complete.

**Return** zero(0) or negative error code.

**Parameters**

- self_p: Driver object.

```c
int adc_convert (struct adc_driver_t *self_p, uint16_t *samples_p, size_t length)
```

Start a synchronous conversion 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.

```c
int adc_convert_isr (struct adc_driver_t *self_p, uint16_t *sample_p)
```

Start a synchronous conversion 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 converted.

**Return** zero(0) or negative error code.

**Parameters**

- self_p: Driver object.
- sample_p: Converted sample.

```c
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**
• \texttt{dev.p}: ADC device to validate.

\section*{Variables}

\begin{verbatim}
struct adc_device_t adc_device[ADCDEVICE_MAX]
\end{verbatim}

\textbf{analog_input_pin} — Analog input pin

Source code: src/drivers/analog_input_pin.h, src/drivers/analog_input_pin.c
Test code: tst/drivers/analog_input_pin/main.c

\section*{Functions}

\begin{verbatim}
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.
  \textbf{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.
  \textbf{Return} zero(0) or negative error code.
  \textbf{Parameters}
  \begin{itemize}
    \item \texttt{self.p}: Driver object to be initialized.
    \item \texttt{dev.p}: Device to use.
  \end{itemize}

int analog_input_pin_read (struct analog_input_pin_t *self_p)
  Read the current value of given pin.
  \textbf{Return} Analog pin value, otherwise negative error code.
  \textbf{Parameters}
  \begin{itemize}
    \item \texttt{self.p}: Driver object.
  \end{itemize}

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.
  \textbf{Return} Analog pin value, otherwise negative error code.
  \textbf{Parameters}
  \begin{itemize}
    \item \texttt{self.p}: Driver object.
  \end{itemize}
\end{verbatim}

\begin{verbatim}
struct analog_input_pin_t
  \#include <analog_input_pin.h>
\end{verbatim}
Public Members

```c
struct adc_driver_t adc
```

**analog_output_pin — Analog output pin**

Source code: `src/drivers/analog_output_pin.h, src/drivers/analog_output_pin.c`

Test code: `tst/drivers/analog_output_pin/main.c`

---

Functions

```c
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.

```c
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.

```c
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.

```c
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.
Public Members

```c
struct pwm_driver_t pwm
```

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.

```c
struct can_frame_t can_rx_buf[8];
struct can_frame_t frame;

/* Initialize and start the CAN controller. */
can_init(&can,
    &can_device[0],
    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/can.h, src/drivers/can.c

Test code: tst/drivers/can/main.c

Defines

```
CAN_SPEED_1000KBPS
CAN_SPEED_500KBPS
CAN_SPEED_250KBPS
```

Functions

```c
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 \texttt{CAN\_SPEED\_}.
  • rxbuf_p: CAN frame reception buffer.
  • size: Size of the reception buffer in bytes.

\textbf{int can\_start (struct can\_driver\_t *self\_p)}
Starts the CAN device using configuration in given driver object.

\textbf{Return  zero(0) or negative error code.}

Parameters
  • self_p: Initialized driver object.

\textbf{int can\_stop (struct can\_driver\_t *self\_p)}
Stops the CAN device referenced by given driver object.

\textbf{Return  zero(0) or negative error code.}

Parameters
  • self_p: Initialized driver object.

\textbf{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.

\textbf{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 \texttt{sizeof(struct can\_frame\_t)}.

\textbf{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.

\textbf{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 \texttt{sizeof(struct can\_frame\_t)}.  

Variables

```c
struct can_device_t can_device[CAN_DEVICE_MAX]
struct can_frame_t
```

Public Members

```c
uint32_t id
uint8_t extended_frame
uint8_t rtr
uint8_t size
struct can_frame_t::*0 can_frame_t::*1
uint8_t u8[8]
uint32_t u32[2]
union can_frame_t::*2 can_frame_t::data
```

chipid — Chip identity

Source code: src/drivers/chipid.h, src/drivers/chipid.c
Test code: tst/drivers/chipid/main.c

Functions

```c
int chipid_read(struct chipid_t *id_p)
```

dac — Digital to analog conversion

Source code: src/drivers/dac.h, src/drivers/dac.c
Test code: tst/drivers/dac/main.c

Functions

```c
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.
```

```c
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 conversion of samples to an analog signal.

Return zero(0) or negative error code.

Parameters
- self_p: Driver object.
- samples: 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 conversion 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 conversion of samples to an analog signal.

Return zero(0) or negative error code.

Parameters
- self_p: Driver object.
- samples: Converted samples.
- length: Length of samples array.

Variables

struct dac_device_t dac_device[DAC_DEVICE_MAX]

ds18b20 — One-wire temperature sensor

Source code: src/drivers/ds18b20.h, src/drivers/ds18b20.c
Test code: tst/drivers/ds18b20/main.c
Defines

DS18B20_FAMILY_CODE

Functions

```c
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 devices 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 temperature conversion on all sensors.
    Return zero(0) or negative error code.
    Parameters
    • self_p: Driver object to be initialized.

int ds18b20_get_temperature (struct ds18b20_driver_t *self_p, const uint8_t *id_p, int *temp_p)
    Get the temperature for given device identity.
    Return zero(0) or negative error code.
    Parameters
    • self_p: Driver object to be initialized.
    • id_p: Device identity.
    • temp_p: Measured temperature in Q4.4 to Q8.4 depending on resolution.

char *ds18b20_get_temperature_str (struct ds18b20_driver_t *self_p, const uint8_t *id_p, char *temp_p)
    Get temperature for given device identity formatted as a string.
    Return temp_p on success, NULL otherwise.
    Parameters
    • self_p: Driver object to be initialized.
    • id_p: Device identity.
    • temp_p: Measured formatted temperature.
```
struct ds18b20_driver_t

Public Members

struct owi_driver_t *owi_p
struct ds18b20_driver_t *next_p

ds3231 — RTC clock

Source code: src/drivers/ds3231.h, src/drivers/ds3231.c
Test code: tst/drivers/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.

#include <ds3231.h>
Public Members

```c
struct i2c_driver_t *i2c_p
```

eeprom_soft — Software EEPROM

Source code: src/drivers/eeprom_soft.h, src/drivers/eeprom_soft.c
Test code: tst/drivers/eeprom_soft/main.c

Functions

```c
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.

```c
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.

```c
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.

```c
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.

struct eeprom_soft_block_t
#include <eeprom_soft.h>

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::@3 eeprom_soft_driver_t:::current
**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`.

```c
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`.

```c
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`.

```c
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`.

```c
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/esp_wifi/softap.h`, `src/drivers/esp_wifi/softap.c`

Test code: `tst/drivers/esp_wifi/softap/main.c`

---

**Functions**

```c
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 connected 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>

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/esp_wifi/station.h, src/drivers/esp_wifi/station.c
Test code: tst/drivers/esp_wifi/station/main.c

### Enums

```c
enum esp_wifi_station_status_t
{
    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

```c
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 configuration or NULL to use DHCP.

```c
int esp_wifi_station_connect (void)
```

Connect the WiFi station to the Access Point (AP).

**Return** zero(0) or negative error code.

```c
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).

denum 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.

denum esp_wifi_dhcp_status_t esp_wifi_station_dhcp_client_status (void)

Get the station DHCP client status.

    Return  Station DHCP client status.

dconst 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/esp_wifi.h, src/drivers/esp_wifi.c

Test code: tst/drivers/esp_wifi/main.c
Enums

enum esp_wifi_op_mode_t

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.

exti — External interrupts

Source code: src/drivers/exti.h, src/drivers/exti.c
Test code: tst/drivers/exti/main.c

**Defines**

EXTI_TRIGER_BOTH_EDGES
    Trigger an interrupt on both rising and falling edges.

EXTI_TRIGER_FALLING_EDGE
    Trigger an interrupt on falling edges.

EXTI_TRIGER_RISING_EDGE
    Trigger an interrupt on both 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_p)
    Initialize given driver object.

    **Return** zero(0) or negative error code.

    **Parameters**
• \texttt{self\_p}: Driver object to be initialized.
• \texttt{dev\_p}: Device to use.
• \texttt{trigger}: One of \texttt{EXTI\_TRIGGER\_BOTH\_EDGES}, \texttt{EXTI\_TRIGGER\_FALLING\_EDGE} or \texttt{EXTI\_TRIGGER\_RISING\_EDGE}.
• \texttt{on\_interrupt}: Function callback called when an interrupt occurs.
• \texttt{arg\_p}: Function callback argument.

\begin{verbatim}
int exti_start (struct exti_driver_t *self)
    Starts the EXTI device using given driver object.
    \textbf{Return} zero(0) or negative error code.
    \textbf{Parameters}
    • \texttt{self\_p}: Driver object.
\end{verbatim}

\begin{verbatim}
int exti_stop (struct exti_driver_t *self)
    Stops the EXTI device referenced by given driver object.
    \textbf{Return} zero(0) or negative error code.
    \textbf{Parameters}
    • \texttt{self\_p}: Driver object.
\end{verbatim}

\begin{verbatim}
int exti_clear (struct exti_driver_t *self)
    Clear the interrupt flag.
    \textbf{Return} zero(0) or negative error code.
    \textbf{Parameters}
    • \texttt{self\_p}: Driver object.
\end{verbatim}

\subsection*{Variables}

\begin{verbatim}
struct exti_device_t exti_device[EXTI\_DEVICE\_MAX]
\end{verbatim}

\textbf{flash — Flash memory}

Source code: src/drivers/flash.h, src/drivers/flash.c
Test code: tst/drivers/flash/main.c

\section*{Functions}

\begin{verbatim}
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.
    \textbf{Return} zero(0) or negative error code.
\end{verbatim}
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 flash memory.

**Return** Number of read bytes 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 flash memory.

**Return** Number of written bytes 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.
- `addr`: Address in flash memory to erase from.
- `size`: Number of bytes to erase.

**Variables**

`struct flash_device_t flash_device[FLASH_DEVICE_MAX]`
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.

This driver is for systems with hardware I2C support. For systems without hardware I2C support the i2c_soft —
Software I2C driver can be used.

Source code: src/drivers/i2c.h, src/drivers/i2c.c
Test code: tst/drivers/i2c/master/main.c

Defines

I2C_BAUDRATE_3_2MBPS
I2C_BAUDRATE_1MBPS
I2C_BAUDRATE_400KBPS
I2C_BAUDRATE_100KBPS

Functions

int i2c_module_init()
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 trans-
mision. 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
**Simba Documentation, Release 15.0.3**

- **self_p**: Driver object to initialize.

**`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/i2c_soft.h, src/drivers/i2c_soft.c

**Test code**: tst/drivers/i2c/master_soft/main.c
Functions

int \texttt{i2c\_soft\_module\_init}(\textit{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.

\textbf{Return} zero(0) or negative error code.

int \texttt{i2c\_soft\_init}\left(\textit{struct} \ i2c\_soft\_driver\_t \ *\textit{self}\_p, \ \textit{struct} \ pin\_device\_t \ *\textit{scl}\_dev\_p, \ \textit{struct} \ pin\_device\_t \ *\textit{sda}\_dev\_p, \ \textit{long} \ \textit{baudrate}, \ \textit{long} \ \textit{max\_clock\_stretching\_us}, \ \textit{long} \ \textit{clock\_stretching\_sleep\_us}\right)

Initialize given driver object.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{self\_p}: Driver object to initialize.
- \textit{scl\_dev\_p}: The I2C clock pin (SCL).
- \textit{sda\_dev\_p}: The I2C data pin (SDA).
- \textit{baudrate}: Bus baudrate.
- \textit{max\_clock\_stretching\_us}: Maximum number of microseconds to wait for the clock stretching to end.
- \textit{clock\_stretching\_sleep\_us}: SCL poll interval in number of microseconds waiting for clock stretching to end.

int \texttt{i2c\_soft\_start}\left(\textit{struct} \ i2c\_soft\_driver\_t \ *\textit{self}\_p\right)

Start given driver object. Enables data reception and transmission, but does not start any transmission. Data transfers are started by calling the \texttt{i2c\_soft\_read()} and \texttt{i2c\_soft\_write()}.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{self\_p}: Driver object to initialize.

int \texttt{i2c\_soft\_stop}\left(\textit{struct} \ i2c\_soft\_driver\_t \ *\textit{self}\_p\right)

Stop given driver object. Disables data reception and transmission.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{self\_p}: Driver object to initialize.

ssize_t \texttt{i2c\_soft\_read}\left(\textit{struct} \ i2c\_soft\_driver\_t \ *\textit{self}\_p, \ \textit{int} \ \textit{address}, \ \textit{void} \ *\textit{buf}\_p, \ \textit{size\_t} \ \textit{size}\right)

Read given number of bytes into given buffer from given slave.

\textbf{Return} Number of bytes read or negative error code.

\textbf{Parameters}

- \textit{self\_p}: Driver object.
- \textit{address}: Slave address to read from.
Simba Documentation, Release 15.0.3

• 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
#include <i2c_soft.h>

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

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/led_7seg_ht16k33.h, src/drivers/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
#include <led_7seg_ht16k33.h>
```
Public Members

```c
struct i2c_soft_driver_t *i2c_p
int i2c_addr
uint8_t buf[5]
```

**mcp2515 — CAN BUS chipset**

Source code: src/drivers/mcp2515.h, src/drivers/mcp2515.c
Test code: tst/drivers/mcp2515/main.c

---

**Defines**

```
MCP2515_SPEED_1000KBPS
MCP2515_SPEED_500KBPS
MCP2515_MODE_NORMAL
MCP2515_MODE_LOOPBACK
```

**Functions**

```c
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.

```c
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
nrf24l01 — Wireless communication

Source code: src/drivers/nrf24l01.h, src/drivers/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 flag device.
• 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
#include <nrf24l01.h>

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/owi.h, src/drivers/owi.c
Test code: tst/drivers/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.

```c
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.

```c
struct owi_device_t
```

**Public Members**

- `uint8_t id[8]`

```c
struct owi_driver_t
```

**Public Members**

- `struct pin_driver_t pin`
- `struct owi_device_t *devices_p`
- `size_t nmemb`
- `size_t len`

**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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_mode &lt;pin&gt; &lt;mode&gt;</td>
<td>Set the mode of the pin &lt;pin&gt; to &lt;mode&gt;, where &lt;mode&gt; is one of output and input.</td>
</tr>
<tr>
<td>read &lt;pin&gt;</td>
<td>Read current input or output value of the pin &lt;pin&gt;, high or low is printed.</td>
</tr>
<tr>
<td>write &lt;pin&gt; &lt;value&gt;</td>
<td>Write the value &lt;value&gt; to pin &lt;pin&gt;, where &lt;value&gt; is one of high and low.</td>
</tr>
</tbody>
</table>

Example output from the shell:
### Defines

**PIN_OUTPUT**

**PIN_INPUT**

Configure the pin as an input pin.

### 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.

**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 `PIN_INPUT` or `PIN_OUTPUT`.

**int pin_write (struct pin_driver_t *self_p, int value)**

Write given value to given pin.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Driver object.
- `value`: 1 for high and 0 for low output.

---

Source code: src/drivers/pin.h, src/drivers/pin.c

Test code: tst/drivers/pin/main.c

---

1.6. Library Reference
int **pin_read** *(struct pin_driver_t *self_p)*
Read the current value of given pin.

**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).

**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.

**Return** zero(0) or negative error code.

**Parameters**
- **self_p**: Driver object.
- **mode**: New pin mode.

static int **pin_device_set_mode** *(const struct pin_device_t *dev_p, int mode)*
Pin device mode to set. One of **PIN_INPUT** or **PIN_OUTPUT**.

**Return** zero(0) or negative error code.

**Parameters**
- **self_p**: Pin device.
- **mode**: New pin mode.

static int **pin_device_read** *(const struct pin_device_t *dev_p)*
Read the value of given pin device.

**Return** 1 for high and 0 for low input, otherwise negative error code.

**Parameters**
- **self_p**: Pin device.

static int **pin_device_write_high** *(const struct pin_device_t *dev_p)*
Write high to given pin device.

**Return** zero(0) or negative error code.

**Parameters**
- **self_p**: Pin device.

static int **pin_device_write_low** *(const struct pin_device_t *dev_p)*
Write low to given pin device.
Return zero(0) or negative error code.

Parameters

• self_p: Pin device.

int pin_is_valid_device (struct pin_device_t *dev_p)
Check if given pin device is valid.

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]

pwm — Pulse width modulation

Source code: src/drivers/pwm.h, src/drivers/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()`.

```c
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/pwm_soft.h, src/drivers/pwm_soft.c
Test code: tst/drivers/pwm_soft/main.c

**Functions**

```c
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()`.
```

```c
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, otherwise 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.
```

```c
long pwm_soft_get_frequency (void)
  Get current frequency.

  **Return**  Current frequency in Hertz.
```

```c
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.
```

```c
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.
Public Members

```c
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/random.h, src/drivers/random.c
Test code: tst/drivers/random/main.c

Functions

```c
int random_module_init (void)
uint32_t random_read (void)
```

Read a random number from the hardware.

**Return** Read random number.

sd — Secure Digital memory

Source code: src/drivers/sd.h, src/drivers/sd.c
Test code: tst/drivers/sd/main.c

Defines

```c
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
```c
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

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 using the \textit{i2c\_soft} — \textit{Software I2C} driver to communicate with the SHT3x sensor, not the \textit{i2c} — \textit{I2C} 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/sht3xd.h, src/drivers/sht3xd.c

Defines

\textbf{SHT3X\_DIS\_I2C\_ADDR\_A}
SHT3x-DIS default I2C address.

\textbf{SHT3X\_DIS\_I2C\_ADDR\_B}
SHT3x-DIS alternate I2C address.

\textbf{MEASUREMENT\_DURATION\_HIGH\_MS}
Max measurement time for high repeatability.

Functions

\textbf{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.

\textbf{Return} zero(0) or negative error code.

\textbf{int sht3xd\_init (struct sht3xd\_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 sensor.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

\begin{itemize}
\item \textbf{self\_p}: Driver object to be initialize.
\item \textbf{i2c\_p}: The I2C driver pointer.
\item \textbf{i2c\_addr}: The address of the SHT3x-D. Probably SHT3X\_DIS\_I2C\_ADDR\_A.
\end{itemize}

\textbf{int sht3xd\_start (struct sht3xd\_driver\_t *self\_p) }
Start the driver.

This verify the sensor is present.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

\begin{itemize}
\item \textbf{self\_p}: Driver object.
\end{itemize}

\textbf{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`: Temperature in Celsius, or NULL.
- `humid_p`: Relative Humidity, or NULL.

```c
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.

```c
struct sht3xd_driver_t
#include <sht3xd.h>
```

Public Members

```c
struct i2c_soft_driver_t *i2c_p
int i2c_addr
uint32_t serial
```

spi — Serial Peripheral Interface

Source code: src/drivers/spi.h, src/drivers/spi.c

---

Defines

```c
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 owns 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)
Simultaneous 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/uart.h, src/drivers/uart.c
Test code: tst/drivers/uart/main.c

Defines

textured _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.

textured _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**

```c
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.
}
```

```c
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.
}
```

```c
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.
}
```

```c
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.
}
```

```c
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_device_start (struct uart_device_t *dev_p, long baudrate)

Starts the UART device using given configuration. 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/uart_soft.h, src/drivers/uart_soft.c

Defines

\texttt{uart\_soft\_read} (self\_p, buf\_p, size)
Read data from the UART.

\textbf{Return} Number of received bytes or negative error code.

\textbf{Parameters}

\begin{itemize}
  \item self\_p: Initialized driver object.
  \item buf\_p: Buffer to read into.
  \item size: Number of bytes to receive.
\end{itemize}

\texttt{uart\_soft\_write} (self\_p, buf\_p, size)
Write data to the UART.

\textbf{Return} number of sent bytes or negative error code.

\textbf{Parameters}

\begin{itemize}
  \item self\_p: Initialized driver object.
  \item buf\_p: Buffer to write.
  \item size: Number of bytes to write.
\end{itemize}

Functions

\texttt{int uart\_soft\_init} (struct \texttt{uart\_soft\_driver\_t} *self\_p, struct \texttt{pin\_device\_t} *tx\_dev\_p, struct \texttt{pin\_device\_t} *rx\_dev\_p, struct \texttt{exti\_device\_t} *rx\_exti\_dev\_p, int baudrate, void *rxbuf\_p, size\_t size)
Initialize driver object from given configuration.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

\begin{itemize}
  \item self\_p: Driver object to be initialized.
  \item tx\_dev\_p: TX pin device.
  \item rx\_dev\_p: RX pin device.
  \item rx\_exti\_dev\_p: RX pin external interrupt device.
  \item baudrate: Baudrate.
  \item rxbuf\_p: Reception buffer.
  \item size: Reception buffer size.
\end{itemize}

\texttt{struct uart\_soft\_driver\_t}

\texttt{#include <uart\_soft.h>}

1.6. Library Reference 273
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/usb.h, src/drivers/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
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::clear_feature
uint16_t zero0
uint16_t zero1

struct usb_setup_t::get_configuration
uint8_t descriptor_index
uint8_t descriptor_type
uint16_t language_id

struct usb_setup_t::get_descriptor
uint16_t device_address
uint16_t zero

struct usb_setup_t::set_address
uint16_t configuration_value

struct usb_setup_t::set_configuration
uint16_t value


```c
uint16_t index
struct usb_setup_t::@20::@26 usb_setup_t::base
union usb_setup_t::@20 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 numInterfaces
uint8_t configuration_value
uint8_t configuration
```
struct usb_descriptor_interfacedata_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
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/usb/device/class/cdc.h, src/drivers/usb/device/class/cdc.c
Test code: tst/drivers/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
    #include <cdc.h>

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 cdc_line_info_t line_info
struct chan_t chout
struct queue_t chin

1.6. Library Reference
Functions

int usb_device_module_init (void)

int usb_device_init(struct usb_device_driver_t * self_p, struct usb_device_t * dev_p, struct usb_device_driver_base_t ** drivers_pp, int drivers_max, FAR const union usb_descriptor_t ** descriptors_pp)

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/usb/host/class/hid.h, src/drivers/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**

```c
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.

```c
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.

```c
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.

**Public Members**

```c
struct usb_host_class_hid_device_t
```

- `buf[1]`
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::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/usb/host/class/mass_storage.h, src/drivers/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
#include <mass_storage.h>

Public Members

uint8_t buf[1]

struct usb_host_class_mass_storage_driver_t
Public Members

```c
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::*19 usb_host_class_mass_storage_driver_t device_driver
```

Source code: src/drivers/usb_host.h, src/drivers/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

```c
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.

```c
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.

```c
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.

```c
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.

```c
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.

```c
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.

```c
#include <usb_host.h> An USB device as seen by the host.
```

**Public Members**

```c
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::descriptor
struct 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]
```

```c
#include <usb_host.h> Used to find a device driver.
```

**Public Members**

```c
int (*supports)(struct usb_host_device_t *)
int (*enumerate)(struct usb_host_device_t *)
struct usb_host_device_driver_t *next_p
```

**watchdog — Hardware watchdog**

Source code: src/drivers/watchdog.h, src/drivers/watchdog.c
Typedefs

typedef void (*watchdog_isr_fn_t)(void)

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.
    • 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.

ws2812 — NeoPixels

Source code: src/drivers/ws2812.h, src/drivers/ws2812.c

Defines

WS2812_PIN_DEVICES_MAX
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 num-
ber_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
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.

```c
/* 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

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 argument 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 \texttt{fat16\_file\_open} \( \text{(struct \texttt{fat16\_t}\ *self\_p,}\ \text{struct} \ \texttt{fat16\_file\_t}\ *file\_p,}\ \text{const}\ \text{char}\ \text{*path\_p},\ \text{int}\ \text{oflag}) \)

Open a file by file path and mode flags.

Return \( 0 \) or negative error code.

Parameters

- \( \text{self\_p} \): FAT16 object.
- \( \text{file\_p} \): File object to be initialized.
- \( \text{path\_p} \): A valid 8.3 DOS name for a file path.
- \( \text{oflag} \): mode of file open (create, read, write, etc).

int \texttt{fat16\_file\_close} \( \text{(struct}\ \texttt{fat16\_file\_t}\ *file\_p)} \)

Close a file and force cached data and directory information to be written to the media.

Return \( 0 \) or negative error code.

Parameters

- \( \text{file\_p} \): File object.

ssize_t \texttt{fat16\_file\_read} \( \text{(struct}\ \texttt{fat16\_file\_t}\ *file\_p,}\ \text{void}\ \text{*buf\_p,}\ \text{size\_t}\ \text{size)} \)

Read data to given buffer with given size from the file.

Return Number of bytes read or EOF(-1).

Parameters

- \( \text{file\_p} \): File object.
- \( \text{buf\_p} \): Buffer to read into.
- \( \text{size} \): number of bytes to read.

ssize_t \texttt{fat16\_file\_write} \( \text{(struct}\ \texttt{fat16\_file\_t}\ *file\_p,}\ \text{const}\ \text{void}\ \text{*buf\_p,}\ \text{size\_t}\ \text{size)} \)

Write data from buffer with given size to the file.

Return Number of bytes written or EOF(-1).

Parameters

- \( \text{file\_p} \): File object.
- \( \text{buf\_p} \): Buffer to write.
- \( \text{size} \): number of bytes to write.

int \texttt{fat16\_file\_seek} \( \text{(struct}\ \texttt{fat16\_file\_t}\ *file\_p,}\ \text{int}\ \text{pos,}\ \text{int}\ \text{whence)} \)

Sets the file’s read/write position relative to mode.

Return \( 0 \) or negative error code.

Parameters

- \( \text{file\_p} \): File object.
- \( \text{pos} \): New position in bytes from given mode.
- \( \text{whence} \): Absolute, relative or from end.
ssize_t fat16_fileTell (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 precisely 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::@30 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):

Public Members

```c
uint16_t as_uint16
uint16_t day
uint16_t month
uint16_t year
```

```c
struct fat16_date_t::bits
```

```c
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

```c
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.

```c
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.

```c
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.

```c
unsigned begin_cylinder_high
```

High bits cylinder for first block in partition.

```c
uint8_t begin_cylinder_low
```

Combine beginCylinderLow with beginCylinderHigh. Legal values are 0-1023. Only used in old PC BIOS.

```c
uint8_t type
```

Partition type. See defines that begin with PART_TYPE_ for some Microsoft partition types.

```c
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.

```c
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.

```c
unsigned end_cylinder_high
```

High bits of end cylinder.

```c
uint8_t end_cylinder_low
```

Combine endCylinderLow with endCylinderHigh. Legal values are 0-1023. Only used in old PC BIOS.

```c
uint32_t first_sector
```

Logical block address of the first block in the partition.

```c
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 hidden_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

```c
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
```

```c
struct mbr_t
    #include <fat16.h> Master Boot Record. The first block of a storage device that is formatted with a MBR.
```

Public Members

```c
uint8_t codeArea[440]
    Code Area for master boot program.

uint32_t diskSignature
    Optional Windows NT 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
```

```c
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

1.6. Library Reference 303
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 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 debug 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/`. 

1.6. Library Reference 305
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filesystems/list</td>
<td>Print a list of all registered file systems.</td>
</tr>
<tr>
<td>list [&lt;folder&gt;]</td>
<td>Print a list of all files and folders in given folder.</td>
</tr>
<tr>
<td>read &lt;file&gt;</td>
<td>Read from given file.</td>
</tr>
<tr>
<td>write &lt;file&gt; &lt;data&gt;</td>
<td>Create and write to a file. Overwrites existing files.</td>
</tr>
<tr>
<td>append &lt;file&gt; &lt;data&gt;</td>
<td>Append data to an existing file.</td>
</tr>
<tr>
<td>counters/list</td>
<td>Print a list of all registered counters.</td>
</tr>
<tr>
<td>counters/reset</td>
<td>Rest all counters to zero.</td>
</tr>
<tr>
<td>parameters/list</td>
<td>Print a list of all registered parameters.</td>
</tr>
</tbody>
</table>

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%      
$ filesystems/fs/write tmp/foo.txt "Hello " 
$ filesystems/fs/append tmp/foo.txt world!  
Hello world! 
$ filesystems/fs/list tmp  
xxxx-xx-xx xx-xx 12 foo.txt 
$ filesystems/fs/counters/list  
NAME           VALUE                  
/your/counter   0000000000000034   
/my/counter     0000000000000002   
$ filesystems/fs/counters/reset 
$ filesystems/fs/counters/list  
NAME           VALUE                  
/your/counter   0000000000000000   
/my/counter     0000000000000000   
$ filesystems/fs/parameters/list  
NAME           VALUE                  
/foo/bar        -2                    
```

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 current position indicator.

**FS SEEK CUR**

The offset is relative to the end of the file.

**FS SEEK END**

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**

```c
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 arguments in argv.
- **argv**: An array of arguments.
- **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.

```c
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.
• dst_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)
Create a directory with given path.

Return zero(0) or negative error code.

Parameters

- path_p: The path of the directory 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 happened, or negative error code.

Parameters

- path_p: Absolute or relative path to auto-complete.
void \texttt{fs\_split} (char *\textit{buf\_p}, char **\textit{path\_pp}, char **\textit{cmd\_pp})

Split buffer into path and command inplace.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{buf\_p}: Buffer to split.
- \textit{path\_pp}: Path or NULL if no path was found.
- \textit{cmd\_pp}: Command or empty string.

void \texttt{fs\_merge} (char *\textit{path\_p}, char *\textit{cmd\_p})

Merge path and command previously split using \texttt{fs\_split}().

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{path\_p}: Path from split.
- \textit{cmd\_p}: Command from split.

\textbf{int} \texttt{fs\_filesystem\_init\_generic} (struct \textit{fs\_filesystem\_t} *\textit{self\_p}, const char *\textit{name\_p}, struct \textit{fs\_filesystem\_operations\_t} *\textit{ops\_p})

Initialize given generic file system.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{self\_p}: File system to initialize.
- \textit{name\_p}: Path to register.
- \textit{ops\_p}: File system function callbacks.

\textbf{int} \texttt{fs\_filesystem\_register} (struct \textit{fs\_filesystem\_t} *\textit{self\_p})

Register given file system. Use the functions \texttt{fs\_open()}, \texttt{fs\_read()}, \texttt{fs\_write()}, \texttt{fs\_close()}, \texttt{fs\_seek()}, \texttt{fs\_tell()} and \texttt{fs\_read\_line()} to access files in a registerd file system.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{self\_p}: File system to register.

\textbf{int} \texttt{fs\_filesystem\_deregister} (struct \textit{fs\_filesystem\_t} *\textit{self\_p})

Deregister given file system.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

- \textit{self\_p}: File system to deregister.

\textbf{int} \texttt{fs\_command\_init} (struct \textit{fs\_command\_t} *\textit{self\_p}, far\_string\_t \textit{path\_p}, \textit{fs\_callback\_t} \textit{callback}, void *\textit{arg\_p})

Initialize given command.

\textbf{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.

```c
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::@32::@34 fs_filesystem_t::generic`
- `union fs_filesystem_t::@32 fs_filesystem_t::fs`
- `union fs_filesystem_t::@33 fs_filesystem_t::config`
- `struct fs_filesystem_t *next_p`

```c
struct fs_file_t
```

**Public Members**

- `struct fs_filesystem_t *filesystem_p`
- `union fs_file_t::@35 fs_file_t::u`

```c
struct fs_stat_t
```

```c
#include <fs.h> Path stats.
```

**Public Members**

- `uint32_t size`
- `uint8_t type`

```c
struct fs_command_t
```

**Public Members**

- `far_string_t path_p`
- `fs_callback_t callback`
- `void *arg_p`
- `struct fs_command_t *next_p`

```c
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::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
enum spiiffs_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 spiiffs_mount (struct spiiffs_t *self_p, struct spiiffs_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,
spiiffs_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 spiiffs_unmount (struct spiiffs_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 spiiffs_creat (struct spiiffs_t *self_p, const char *path_p, spiiffs_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.
- `ent_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
• \texttt{self\_p}: The file system struct.
• \texttt{old\_path\_p}: Path of file to rename.
• \texttt{new\_path\_p}: New path of file.

\textbf{int32\_t \texttt{spiffs\_errno}} (\textbf{struct spiffs\_t *self\_p})

Returns last error of last file operation.

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

• \texttt{self\_p}: The file system struct.

\textbf{void \texttt{spiffs\_clearerr}} (\textbf{struct spiffs\_t *self\_p})

Clears last error.

\textbf{Return} void.

\textbf{Parameters}

• \texttt{self\_p}: The file system struct.

\textbf{struct spiffs\_dir\_t \texttt{*spiffs\_opendir}} (\textbf{struct spiffs\_t *self\_p, const char \texttt{*name\_p, struct spiffs\_dir\_t \texttt{*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.

\textbf{Parameters}

• \texttt{self\_p}: The file system struct.

• \texttt{name\_p}: The name of the directory.

• \texttt{dir\_p}: Pointer the directory stream to be populated.

\textbf{int32\_t \texttt{spiffs\_closedir}} (\textbf{struct spiffs\_dir\_t *dir\_p})

Closes a directory stream

\textbf{Return} zero(0) or negative error code.

\textbf{Parameters}

• \texttt{dir\_p}: The directory stream to close.

\textbf{struct spiffs\_dirent\_t \texttt{*spiffs\_readdir}} (\textbf{struct spiffs\_dir\_t *dir\_p, struct spiffs\_dirent\_t *ent\_p})

Reads a directory into given spifs\_dirent struct.

\textbf{Return} NULL if error or end of stream, else given dirent is returned.

\textbf{Parameters}

• \texttt{dir\_p}: Pointer to the directory stream.

• \texttt{ent\_p}: The dirent struct to be populated.

\textbf{int32\_t \texttt{spiffs\_check}} (\textbf{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 are 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, SPIFFSUnmount 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.

```c
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.

```c
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.

```c
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.

```c
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

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 security 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
  Values:
    http_server_request_action_get_t = 0
    http_server_request_action_post_t = 1

enum http_server_content_type_t
  Content type.
  Values:
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::@38::@39 http_server_request_t::sec_websocket_key
struct http_server_request_t::@38::@40 http_server_request_t::content_type
long value
struct http_server_request_t::@38::@41 http_server_request_t::content_length
struct http_server_request_t::@38::@42 http_server_request_t::authorization
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::content

struct http_server_listener_t

const char *address_p
int port
const char *name_p
void *buf_p
size_t size
struct http_server_listener_t::stack
struct thrd_t *id_p
struct http_server_listener_t::thrd
struct socket_t socket

struct http_server_connection_t

http_server_connection_state_t state
const char *name_p
void *buf_p
size_t size
struct http_server_connection_t::stack
struct thrd_t *id_p
struct 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**

`#include <http_websocket_client.h>`

**Public Members**

- `struct socket_t` `socket`
- `const char *host_p`
- `int port`

**struct http_websocket_client_t::@49** `http_websocket_client_t::server`

**size_t** `left`

**struct http_websocket_client_t::@50** `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
#include <http_websocket_server.h>

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.

```c
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 checksum of.
- **size**: Size of the buffer.

```c
struct inet_ip_addr_t
#include <inet.h>
```

Public Members

```c
uint32_t number
```
IPv4 address.

```c
struct inet_addr_t
```

Public Members

```c
struct inet_ip_addr_t ip
```
IPv4 address.

```c
uint16_t port
```
Port.

```c
struct inet_if_ip_info_t
#include <inet.h> Interface IP information.
```

Public Members

```c
struct inet_ip_addr_t address
```

```c
struct inet_ip_addr_t netmask
```

```c
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.
Public Members

- uint8_t * `message_p`
- size_t `size`
- int `state`
- int `flags`
- size_t `offset`
- int `next_index`
- struct isotp_t::message

mqtt_client — MQTT client

MQTT is a publish-subscribe-based lightweight messaging protocol.

**Note:** This driver only implements the MQTT protocol, not the transport layer (normally TCP). That has to be set up using channels.

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.

Basic example of initializing MQTT over TCP (error checking left out for brevity).

```c
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);
}
```

```c
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);

mqtt_client_init(&client, 
                  "mqtt_client", 
                  NULL, 
                  &server_sock, 
                  &server_sock, 
                  on_publish,
```
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

**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 negative error code.

**Parameters**

- `client_p`: The client.
- `error`: The number of error that occurred.

** Enums**

typedef enum mqtt_client_state_t

**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)

Establish a connection to the server.

Return zero(0) or negative error code.

Parameters

• self_p: MQTT client.

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_client_t
#include <mqtt_client.h> MQTT client.

Public Members

const char *name_p
struct log_object_t *log_object_p
Simba Documentation, Release 15.0.3

```c
int state
int type
void *data_p
struct mqtt_client_t::message
void *out_p
void *in_p
struct mqtt_client_t::transport
struct queue_t out
struct queue_t in
struct 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

const char *buf_p
size_t size
struct mqtt_application_message_t::topic
const void *buf_p
struct mqtt_application_message_t::payload
mqtt_qos_t qos
```

**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

e num 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_addr_t *ipaddr_p, struct inet_addr_t *netmask_p, struct inet_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

\texttt{network\_interface\_slip\_state_t state}
\begin{itemize}
  \item \texttt{struct pbuf *pbuf\_p}
  \item \texttt{uint8\_t *buf\_p}
  \item \texttt{size\_t size}
\end{itemize}
\texttt{struct network\_interface\_slip\_t::frame}
\texttt{void *chout\_p}
\texttt{struct network\_interface\_t network\_interface}

\texttt{network\_interface\_wifi --- WiFi network interface}

WiFi network interface driver modules:

\texttt{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

\texttt{struct network\_interface\_wifi\_driver\_t network\_interface\_wifi\_driver\_esp\_station}
\texttt{struct network\_interface\_wifi\_driver\_t network\_interface\_wifi\_driver\_esp\_softap}

Espressif 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

\texttt{int network\_interface\_wifi\_module\_init (void)}

Initialize the WiFi network interface module.

\hspace*{1em}Return\hspace{1em}zero(0) or negative error code.

\texttt{int network\_interface\_wifi\_init (struct network\_interface\_wifi\_t *self\_p, const char *name\_p,}
\texttt{struct network\_interface\_wifi\_driver\_t *driver\_p, void *arg\_p,}
\texttt{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 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.

```c
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.

```c
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.

```c
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.

```c
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.

```c
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).

```c
struct network_interface_wifi_t
#include <wifi.h>
```

**Public Members**

```c
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
```

```c
struct network_interface_wifi_driver_t
#include <wifi.h>
```

Driver virtualization callbacks. See the driver/ subfolder for available drivers.

**Public Members**

```c
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/.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Print a list of all registered network interfaces.</td>
</tr>
</tbody>
</table>

Example output from the shell:

```
$ inet/network_interface/list
NAME     STATE ADDRESS  TX BYTES RX BYTES
esp-wlan-ap up 192.168.4.1 - -
est-wlan-sta up 192.168.0.5 - -
```

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. Packages can only be sent and received when the interface is 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/`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ping &lt;remote host&gt;</td>
<td>Ping a remote host by given ip address.</td>
</tr>
</tbody>
</table>

Example output from the shell:

```bash
$ inet/ping/ping 192.168.1.100
Successfully pinged '192.168.1.100' in 10 ms.
```

Source code: `src/inet/ping.h`, `src/inet/ping.c`

Test code: `tst/inet/ping/main.c`

Test coverage: `src/inet/ping.c`

**Functions**

```c
int ping_module_init (void)

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.
```c
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;
inat_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.

```c
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;
inat_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: Number 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::@59::@61::@63 socket_t::common
struct pbuf *pbuf_p
struct inet_addr_t remote_addr
int closed

struct socket_t::@59::@61::@64 socket_t::recvfrom
struct tcp_pcb *pcb_p
struct socket_t::@59::@61::@65 socket_t::accept
union socket_t::@59::@61 socket_t::u
int state
void *args_p
struct thrd_t *thrd_p

struct socket_t::@59::@62 socket_t::cb
struct socket_t::@59 socket_t::input
struct socket_t::@60::@66 socket_t::cb
struct socket_t::@60 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:

```c
/* 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:

```c
/* 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");

/* 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);
```
Defines

SSL_SOCKET_SERVER_SIDE

 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`: Number 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
#include <tftp_server.h>

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

oam

Operations and maintenance 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

```c
int console_module_init (void)
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)
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.

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/.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>List all registered services.</td>
</tr>
<tr>
<td>start &lt;service&gt;</td>
<td>Start given service.</td>
</tr>
<tr>
<td>stop &lt;service&gt;</td>
<td>Stop given service.</td>
</tr>
</tbody>
</table>

Example output from the shell:
$ oam/service/list

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>http_server</td>
<td>running</td>
</tr>
<tr>
<td>ftp_server</td>
<td>stopped</td>
</tr>
<tr>
<td>network_manager</td>
<td>running</td>
</tr>
</tbody>
</table>

$ oam/service/start ftp_server
$ oam/service/stop http_server
$ oam/service/list

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>http_server</td>
<td>stopped</td>
</tr>
<tr>
<td>ftp_server</td>
<td>running</td>
</tr>
<tr>
<td>network_manager</td>
<td>running</td>
</tr>
</tbody>
</table>

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_CONTROL_EVENT_STOP
Service 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
**settings — Persistent application settings**

Settings are stored in a non-volatile memory (NVM). In other words, settings are preserved 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/`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Print a list of the current settings.</td>
</tr>
<tr>
<td>reset</td>
<td>Overwrite the current settings values with their default values (the values defined in the ini-file values).</td>
</tr>
<tr>
<td>read &lt;name&gt;</td>
<td>Read the value of setting &lt;name&gt;.</td>
</tr>
<tr>
<td>write &lt;name&gt; &lt;value&gt;</td>
<td>Write &lt;value&gt; to setting &lt;name&gt;.</td>
</tr>
</tbody>
</table>

Example output from the shell:

```bash
$ 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
$ oam/settings/read value_1
24567
$ oam/settings/write value_1 -5
$ oam/settings/read value_1
-5
$ oam/settings/reset
$ 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
```
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`.

```plaintext
[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.

```c
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

define setting_type_t
    Settings types. Each setting must have 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.

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 (void)

Overwrite all settings with 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. 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.

  Here is an example of using the shell to list and execute debug file system commands.

$$
$ <tab>   
drivers/   
kernel/   
$ kernel/<tab>   
fs/  
sys/  
thrd/   
$ kernel/thrd/list
  NAME   STATE  PRIO  CPU LOGMASK
  main   current 0 0%  0x0f
  idle   ready  127 0%  0x0f
  monitor ready -80 0%  0x0f
$$

  Return  Never returns.

Parameters
  • arg_p: Pointer to the shell argument struct struct shell_t. See the struct definition for a description of it’s content.

struct shell_history_elem_t
  #include <shell.h>
Public Members

```c
struct shell_history_elem_t *next_p
struct shell_history_elem_t *prev_p
char buf[1]
```

struct shell_line_t

Public Members

```c
char buf[CONFIG_SHELL_COMMAND_MAX]
int length
int cursor
```

struct shell_t

Public Members

```c
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::@99::@100 shell_t::heap
struct shell_t::@99 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 the 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.

```c
/* 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.

```c
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.

```shell
> 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
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 |
```

1.6. Library Reference
OK is printed by the shell if the file system command returned \texttt{zero(0)}, otherwise \texttt{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**

```c
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.

```c
struct soam_t
    #include <soam.h>
```

Public Members

```c
int is_printf
uint8_t transaction_id
uint8_t *buf_p
size_t size
ssize_t pos
struct sem_t sem
void *chout_p
uint8_t packet_index
struct soam_t::@101 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/`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/enter</td>
<td>Enter the application.</td>
</tr>
<tr>
<td>application/erase</td>
<td>Erase the application. May not be called from the application about to be erased.</td>
</tr>
<tr>
<td>application/is_valid</td>
<td>Check is there is a valid application in the memory.</td>
</tr>
<tr>
<td>kermit/upload</td>
<td>Upload a upgrade binary file using the Kermit file transfer protocol.</td>
</tr>
<tr>
<td>bootloader/enter</td>
<td>Enter the bootloader.</td>
</tr>
</tbody>
</table>

Example output from the shell:

```
$ oam/upgrade/application/is_valid
yes
```
**HTTP requests**

Five HTTP requests are available. Form the URL by prefixing them with `http://<hostname>/oam/upgrade/`, i.e. `http://<hostname>/oam/upgrade/application/is_valid`.

<table>
<thead>
<tr>
<th>Request</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/enter</td>
<td>GET</td>
<td>Enter the application.</td>
</tr>
<tr>
<td>application/erase</td>
<td>GET</td>
<td>Erase the application. May not be called from the application about to be erased.</td>
</tr>
<tr>
<td>application/is_valid</td>
<td>GET</td>
<td>Check if there is a valid application in the memory.</td>
</tr>
<tr>
<td>upload</td>
<td>POST</td>
<td>Upload a upgrade binary file using the Kermit file transfer protocol.</td>
</tr>
<tr>
<td>bootloader/enter</td>
<td>GET</td>
<td>Enter the bootloader.</td>
</tr>
</tbody>
</table>

**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
$ 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
$ oam/upgrade/application/is_valid
no
$ oam/upgrade/application/erase
$ 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
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)

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 bootloader is forced to enter its main loop instead of calling any valid application.

    Return true(1) if the bootloader 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.

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:

```c
int foo_bar();
int foo_fie();
```

foo_bar() and foo_fie() called in fum.c. Both function calls will call the stubbed version on the respective
function.

```c
int fum_init()
{
    foo_bar();
    foo_fie();
}
```

The stubbed implementations, often defined in the test suite file main.c:

```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
```

**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(struct harness_t *harness_p) {
    /* Return zero(0) when a test case passes. */
    return (0);
}

static int test_failed(struct harness_t *harness_p) {
    /* Return a negative integer when a test case fails. BTASSERT
     * will return -1 when the condition is false. */
    BTASSERT(0);
    return (0);
}

static int test_skipped(struct harness_t *harness_p) {
    /* 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_t harness;
    struct harness testcase_t harness_testcases[] = {
        { test_passed, "test_passed" },
        { test_failed, "test_failed" },
        { test_skipped, "test_skipped" },
        { NULL, NULL }
    };
    sys_start();
    harness_init(&harness);
    harness_run(&harness, harness_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
```
There are plenty of test suites in the tst folder on Github.

Source code: src/debug/harness.h, src/debug/harness.c

**Defines**

```c
#define _ASSERTFMT(fmt, ...)
#define _ASSERTHEX(actual_str, actual, expected_str, expected, size)
#define BTASSERTRM(cond, cond_str, res, msg)
#define BTASSERTR(cond, cond_str, res, ...)
#define BTASSERTN(cond, ...)
#define BTASSERT(res, ...)
#define BTASSERTI(actual, operator, expected)
#define BTASSERTM(actual, expected, size)
#define BTASSERTV(cond, ...)
#define STUB(function)
```

_stub (function)

Stub given function. Used with the make variable STUB to preprocess object file(s).
Typedefs

typedef int (*harness_testcase_cb_t)(struct harness_t *harness_p);

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.

Parameters
  • harness_t: The harness object.

Functions

int harness_init (struct harness_t *self_p);
Initialize given test harness.

Return zero(0) or negative error code.

Parameters
  • self_p: Test harness to initialize.

int harness_run (struct harness_t *self_p, struct harness_testcase_t *testcases_p);
Run given testcases in given test harness.

Return zero(0) or negative error code.

Parameters
  • self_p: Test harness.
  • 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);
Continuously read from the channel and return when given pattern has been read, or when a timeout occurs.

Return Number of bytes read from the channel when match occurred, or negative error code.

Parameters
  • chan_p: Channel to read from.
  • pattern_p: Pattern to wait for.
  • timeout_p: Timeout, or NULL to wait forever.

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.

Return Number of written words or negative error code.

Parameters
  • id_p: Mock id string to write.
  • buf_p: Data for given mock id.
  • size: Buffer size in words.
ssize_t harness_mock_read (const char *id_p, void *buf_p, size_t size)
Read data from mock entry with given id.

Return Number of read words or negative error code.

Parameters
• id_p: Mock id string to read.
• buf_p: Buffer to read into.
• size: Buffer size in words.

struct harness_testcase_t

Public Members

harness_testcase_cb_t callback
const char *name_p

struct harness_t

Public Members

struct uart_driver_t uart

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 argument 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_<uppercase 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.

\[<\text{timestamp}>:<\text{log level}>:<\text{thread name}>:<\text{log object name}>: <\text{message}>\]

Debug file system commands

Three debug file system commands are available, all located in the directory debug/log/.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Print a list of all log objects.</td>
</tr>
<tr>
<td>print &lt;string&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>set_log_mask</td>
<td>Set the log mask to &lt;mask&gt; for log object &lt;object&gt;.</td>
</tr>
</tbody>
</table>

Example output from the shell:

```
$ debug/log/list
  OBJECT NAME  MASK
  default  0x0f
$ debug/log/print "Hello World!"
$ debug/log/set_log_mask default 0x1f
$ debug/log/list
  OBJECT NAME  MASK
  default  0x1f
$ debug/log/print "Hello World!!!"
56:info:main:default: Hello World!!!
```

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:

```c
/* 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.
Simba Documentation, Release 15.0.3

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

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)
Create 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

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).

![Binary Tree Diagram]

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
#include <binary_tree.h>

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

---

**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**

**static uint32_t bits_insert_32 (uint32_t dst, int position, int size, uint32_t src)**

**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.

```c
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.

```c
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.

```c
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.

```c
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.

```c
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 pointer to the start of the array. Only valid if the return value is greater than zero(0).
• `size`: Number of bytes asked for.

```c
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 pointer to the start of the array. Only valid if the return value is greater than zero (0).
- `size`: Number of bytes asked for.

```c
struct circular_buffer_t
#include <circular_buffer.h>
```

**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.

```c
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**

```
#include <fifo.h>
```

---

**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_function_t)(long 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_function_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, long key, void *value_p)
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 hash.
  • value_p: Value to insert for key.

int hash_map_remove (struct hash_map_t *self_p, long key)
Remove given key from hash map.

  Return  zero(0) or negative error code.

  Parameters
  • self_p: Initialized hash map.
  • key: Key to hash.

void *hash_map_get (struct hash_map_t *self_p, long key)
Get value for given key.

  Return  Value for key or NULL if key was not found in the map.

  Parameters
  • self_p: Initialized hash map.
  • key: Key to hash.
struct hash_map_entry_t

    Public Members

    struct hash_map_entry_t *next_p
    long key
    void *value_p

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_function_t hash

list — Abstract lists

Source code: src/collections/list.h

---

Defines

LIST_SL_INIT (list_p)
Initialize given singly linked list object.

Parameters

    • list_p: List object to initialize.

LIST_SL_INIT_STRUCT

LIST_SL_PEEK_HEAD (list_p, element_pp)
Peek at the first element in the list.

Parameters

    • list_p: List object.
    • element_pp: First element of the list.
LIST_SL_ADD_HEAD(list_p, element_p)
Add given element to the beginning of given list.

Parameters
• list_p: List object.
• element_p: Element to add.

LIST_SL_ADD_TAIL(list_p, element_p)
Add given element to the end of given list.

Parameters
• list_p: List object.
• element_p: Element to add.

LIST_SL_REMOVE_HEAD(list_p, element_pp)
Get the first element of given list and then remove it from given list.

Parameters
• list_p: List object.
• element_pp: First element of the list.

LIST_SL_ITERATOR_INIT(iterator_p, list_p)
Initialize given iterator object.

Parameters
• iterator_p: Iterator to initialize.
• list_p: List object to iterate over.

LIST_SL_ITERATOR_NEXT(iterator_p, element_pp)
Get the next element from given iterator object.

Parameters
• iterator_p: Iterator object.
• element_pp: Next element of the list.

LIST_SL_REMOVE_ELEM(list_p, iterator_p, element_p, iterator_element_p, previous_element_p)
Remove given element from given list.

Parameters
• list_p: List object.
• iterator_p: Used internally.
• element_p: Element to remove.
• iterator_element_p: Used internally.
• previous_element_p: Used internally.
struct list_next_t
  #include <list.h>

Public Members

void *next_p

struct list_singly_linked_t

Public Members

void *head_p
void *tail_p

struct list_sl_iterator_t

Public Members

void *next_p

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
|-------------------------|================|----------|
```

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.

```
begin alloc 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.

**Return** zero(0) or negative error code.

**Parameters**

- `self_p`: Circular heap to initialize.
- `buf_p`: Memory buffer.
- `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 on failure.

**Parameters**
Simba Documentation, Release 15.0.3

- **self_p**: Circular heap to allocate from.
- **size**: Number of bytes to allocate.

```c
int circular_heap_free(struct circular_heap_t *self_p, void *buf_p)
```
Free the oldest allocated buffer.

**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.

```c
struct circular_heap_t
#include <circular_heap.h>
```

**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`

**Functions**

```c
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.
void *heap_alloc (struct heap_t *self_p, size_t size)
Allocate a buffer of given size from given heap.

Return  Pointer to allocated buffer, or NULL on failure.

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 share count by once 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

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

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
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

```plaintext
; 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.
- `length`: 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
  #include <configfile.h>

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)

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

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

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 pattern, 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.

```c
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.

```c
struct re_group_t
```

**Public Members**

```c
const char *buf_p
size_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**

```c
int std_module_init (void)
```

```c
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**: 1 for long or nothing
*specifier: c, s, d, i, u, x or f

**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.

```c
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, or 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.

```c
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.

```c
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, 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.

```c
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.

```c
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 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.

```c
ssize_t std_fprintf(void *chan_p, far_string_t fmt_p, ...)
```
Format and print data to channel. The output is not null terminated.

See `std_sprintf()` for 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.

```c
ssize_t std_vfprintf(void *chan_p, far_string_t fmt_p, va_list *ap_p)
```
Format and print data to channel. The output is not null terminated.

See `std_sprintf()` for 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.

```c
const char *std_strtol(const char *str_p, long *value_p)
```
Convert string to integer.

**Return** Pointer to the next byte or NULL on failure.

**Parameters**
- `str_p`: Integer string.
- `value_p`: Integer value.

```c
const char *std_strtod(const char *str_p, double *value_p)
```
Convert string to double.
Return  Pointer to the next byte or NULL on failure.

Parameters
  • str_p: Double string.
  • value_p: Double value.

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.

cchar *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.

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

```c
int base64_encode (char *dst_p, const void *src_p, size_t size)

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**

```c
enum json_type_t
```

**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.

```c
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 an 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.

```c
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.

```c
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.

```c
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.

```c
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.

```c
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.

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

Functions

uint32_t crc_32 (uint32_t crc, const void *buf_p, size_t size)
    Calculate a 32 bits crc using the polynomial \( x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+x^0 \).

    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.

```c
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.

```c
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.

```c
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.
- **polynomial**: 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.

**Public Members**

- uint8_t **buf**[64]
- uint32_t **size**

   **struct sha1_t::block**
   - uint32_t **h**[5]
   - uint64_t **size**

**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_INSTRUMENT
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.

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
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

```c
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
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.

**arduinonano — Arduino Nano**

Source code: `src/boards/arduinonano/board.h`, `src/boards/arduinonano/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
pwm_d3_dev
pwm_d9_dev
pwm_d10_dev
pwm_d11_dev
adc_0_dev
i2c_0_dev
```
## Functions

```c
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

```c
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
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.

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
pin_led_dev
flash_0_dev
Functions

```c
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

```c
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
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
Functions

```c
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

```c
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**

```c
#define pin_gpio00_dev
#define pin_gpio01_dev
#define pin_gpio02_dev
#define pin_gpio03_dev
#define pin_gpio04_dev
#define pin_gpio05_dev
#define pin_gpio06_dev
#define pin_gpio07_dev
#define pin_gpio08_dev
#define pin_gpio09_dev
#define pin_gpio10_dev
#define pin_gpio11_dev
#define pin_gpio12_dev
#define pin_gpio13_dev
#define pin_gpio14_dev
#define pin_gpio15_dev
#define pin_gpio16_dev
#define pin_gpio17_dev
#define pin_gpio18_dev
#define pin_gpio19_dev
#define pin_gpio21_dev
#define pin_gpio22_dev
```

---

1.6. Library Reference 439
Functions

```c
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
硬件参考：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
Functions

```c
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
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.

photon — Photon

Source code: src/boards/photon/board.h, src/boards/photon/board.c
Hardware reference: Photon
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
Functions

```c
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.
```

Define

```
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
Functions

```c
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

```c
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
Simba Documentation, Release 15.0.3

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_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.

mcus

The Micro Controller Units (MCU:s) supported by Simba.

The MCU:s on Github.

atmega2560 — AT Mega2560

Source code: src/mcus/atmega2560/mcu.h

Defines

PIN_DEVICE_MAX
EXTI_DEVICE_MAX
SPI_DEVICE_MAX
UART_DEVICE_MAX
defines

PIN_DEVICE_MAX
EXTIDEVICE_MAX
SPI_DEVICE_MAX
UART_DEVICE_MAX
PWM_DEVICE_MAX
ADC_DEVICE_MAX
I2CDEVICE_MAX
USART0_TX_vect
USART0_RX_vect
USART0_UDRE_vect

atmega32u4 — ATMega32u4

Source code: src/mcus/atmega32u4/mcu.h

1.6. Library Reference
USART0_UDRE_vect
UCSZ00
UCSZ01
UCSZ02
UPM00
UPM01
USBS0
U2X0
UPE0
DOR0
FE0
TXC0
RXCIE0
RXEN0
TXEN0
UDRE0
UDRIE0
TXCIE0

esp32 — Esp32

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

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

sam3x8e — SAM3X8E

Source code: src/mcus/sam/mcu.h

1.6. Library Reference
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
CANDEVICE_MAX
FLASH_DEVICE_MAX

stm32f303vc — STM32F303VC

Source code: src/mcus/stm32f303vc/mcu.h

License

The Simba project as a whole is licensed under the following MIT license:

The MIT License (MIT)

Copyright (c) 2014-2017, 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:

1.7. License
• 3pp/*: Various licenses.
• src/filesystems/fat16/*: GNU LGPL License

Videos

#6 Simba: CAN client-server test suite on Nano32 (ESP32) and Arduino Due.
Transmit CAN frames between a Nano32 and an Arduino Due.

#5 Simba: Room temperature (DS18B20).
Read and print the room temperature measured with a DS18B20 sensor.

#4 Simba: Hello world.
This application prints “Hello world!” to standard output.

#3 Simba: Analog read.
Read the value of an analog pin periodically once every second and print the read value to standard output.

#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 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.

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.

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
Wingfence

A BWF for a home made robot mower.

Github: https://github.com/wingstar74/wingfence
• **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.
To ensure high code quality each module is tested extensively by many test suites. See *Testing* for details.
Design goals

- Rapid development.
- Clean interfaces.
- Small memory footprint.
- No dynamic memory allocation.
- Portability.
CHAPTER 5

Indices and tables

• genindex
• modindex
• search
a
adc, 220
analog_input_pin, 222
analog_output_pin, 223
arduino_due, 422
arduino_mega, 427
arduino_nano, 430
arduino_pro_micro, 431
arduino_uno, 432
assert, 178
atmega2560, 452
atmega328p, 453
atmega32u4, 453

b
base64, 411
binary_tree, 389
bits, 391
bus, 203

c
can, 224
chan, 205
chipid, 226
circular_buffer, 391
circular_heap, 398
color, 402
cconfigfile, 403
console, 362
crc, 416
d
dac, 226
ds18b20, 227
ds3231, 229
e
eeprom_soft, 230
emacs, 405
errno, 179
esp01, 433
esp12e, 434
esp32, 454
esp32_devkitc, 435
esp8266, 455
esp_wifi, 232
esp_wifi_softap, 232
esp_wifi_station, 234
event, 211
exti, 237

f
fat16, 294
fifo, 393
flash, 238
fs, 305

h
harness, 380
hash_map, 394
heap, 400
http_server, 330
http.websocket_client, 334
http.websocket_server, 336
huzzah, 437

i
i2c, 240
i2c_soft, 242
inet, 337
isotp, 338

j
json, 412

l
led_7seg_ht16k33, 244
linux, 455
list, 396
log, 384
m
maple Esp32, 439
mcp2515, 247
midi, 419
mqtt_client, 340
n
nano32, 441
network_interface, 344
network_interface_driver_esp, 346
network_interface_slip, 344
network_interface_wifi, 346
nodemcu, 442
nrf24l01, 249
nvm, 363
o
owi, 251
p
photon, 443
pin, 252
ping, 351
pwm, 255
pwm_soft, 257
q
queue, 213
r
random, 260
re, 405
rwlock, 216
s
sam3x8e, 455
sd, 260
sem, 218
service, 364
settings, 367
shal, 417
shell, 370
sht3xd, 265
soam, 373
socket, 351
spc56d4011, 456
spc56ddiscovery, 444
spi, 267
spiffs, 317
ssl, 357
std, 407
stm32f100rb, 456
stm32f205rg, 456
stm32f303vc, 457
stm32f3discovery, 446
stm32vldiscovery, 449
sys, 186
t
tftp_server, 361
thrd, 190
time, 197
timer, 199
types, 201
u
uart, 270
uart_soft, 273
upgrade, 376
usb, 274
usb_device, 282
usb_device_class_cdc, 282
usb_host, 285
usb_host_class_hid, 285
usb_host_class_mass_storage, 287
w
watchdog, 291
wemos_d1_mini, 451
ws2812, 292
Symbols

_ASSERTFMT (C macro), 382
_ASSERTHEX (C macro), 382

A
adc (module), 220
adc_0_dev (C macro), 426, 429, 430, 432–434, 436–438, 440, 442, 443, 452
adc_1_dev (C macro), 436, 440, 442
adc_async_convert (C++ function), 221
adc_async_wait (C++ function), 221
adc_convert (C++ function), 221
adc_convert_isr (C++ function), 221
adc_device (C++ member), 222
ADC_DEVICE_MAX (C macro), 453–455
adc_init (C++ function), 220
adc_is_valid_device (C++ function), 221
adc_module_init (C++ function), 220
ADC_PINS_MAX (C macro), 434, 436, 437, 440, 442, 452
ADC_REFERENCE_VCC (C macro), 220
analog_input_pin (module), 222
analog_input_pin_init (C++ function), 222
analog_input_pin_module_init (C++ function), 222
analog_input_pin_read (C++ function), 222
analog_input_pin_read_isr (C++ function), 222
analog_input_pin_t (C++ class), 222
anallog_input_pin_t::adc (C++ member), 222
anallog_input_pin_module (module), 223
anallog_input_pin_init (C++ function), 223
anallog_input_pin_module_init (C++ function), 223
anallog_input_pin_read (C++ function), 223
anallog_input_pin_read_isr (C++ function), 223
anallog_input_pin_t (C++ class), 223
anallog_input_pin_t::pwm (C++ member), 224
anallog_input_pin_module (module), 223
arduino_due (module), 422
arduino_mega (module), 427
arduino_nano (module), 430
arduino_pro_micro (module), 431
arduino_uno (module), 432
ASSERT (C macro), 178
assert (module), 178
ASSERTN (C macro), 178
ASSERTRN (C macro), 178
ASSERTRV (C macro), 178
ASSERTRNV (C macro), 179
ASSERTRNRN (C macro), 179
assert (module), 178
atmega2560 (module), 452
atmega328p (module), 453
atmega32u4 (module), 453

B
base64 (module), 411
base64_decode (C++ function), 412
base64_encode (C++ function), 412
binary_tree (module), 389
binary_tree_delete (C++ function), 390
binary_tree_init (C++ function), 389
binary_tree_insert (C++ function), 389
binary_tree_node_t (C++ class), 390
binary_tree_node_t::height (C++ member), 390
binary_tree_node_t::key (C++ member), 390
binary_tree_node_t::left_p (C++ member), 390
binary_tree_node_t::right_p (C++ member), 390
binary_tree_print (C++ function), 390
binary_tree_search (C++ function), 390
binary_tree_t (C++ class), 390
bpb_t (C++ class), 300
bpb_t::bytes_per_sector (C++ member), 301
bpb_t::fat_count (C++ member), 301
bpb_t::head_count (C++ member), 301
bpb_t::hiddened_sectors (C++ member), 301
bpb_t::media_type (C++ member), 301
bpb_t::reserved_sector_count (C++ member), 301
bpb_t::root_dir_entry_count (C++ member), 301
bpb_t::sectors_per_cluster (C++ member), 301
bpb_t::sectors_per_fat (C++ member), 301
bpb_t::sectors_per_track (C++ member), 301
bpb_t::total_sectors_large (C++ member), 301
bpb_t::total_sectors_small (C++ member), 301
BTASSERT (C macro), 382
BTASSERTI (C macro), 382
BTASSERTM (C macro), 382
BTASSERTN (C macro), 382
BTASSERTR (C macro), 382
BTASSERTRM (C macro), 382
BTASSERTV (C macro), 382
bus (module), 203
bus_attach (C++ function), 204
bus_detatch (C++ function), 204
bus_init (C++ function), 203
bus_listener_init (C++ function), 203
bus_listener_t (C++ class), 205
bus_listener_t::base (C++ member), 205
bus_listener_t::id (C++ member), 205
bus_listener_t::next_p (C++ member), 205
bus_module_init (C++ function), 203
bus_t (C++ class), 204
bus_t::listeners (C++ member), 205
bus_t::rwlock (C++ member), 205
bus_write (C++ function), 204
C
can (module), 224
can_0_dev (C macro), 449
can_device (C++ member), 226
CAN_DEVICE_MAX (C macro), 454–457
can_frame_t (C++ class), 226
can_frame_t::extended_frame (C++ member), 226
can_frame_t::id (C++ member), 226
can_frame_t::rtr (C++ member), 226
can_frame_t::size (C++ member), 226
can_frame_t::u32 (C++ member), 226
chan (module), 205
chan_control (C++ function), 208
chan_control_fn_t (C++ type), 206
CHAN_CONTROL_LOG_BEGIN (C macro), 205
CHAN_CONTROL_LOG_END (C macro), 205
chan_control_null (C++ function), 210
chan_is_polled_isr (C++ function), 209
chan_list_add (C++ function), 209
chan_list_destroy (C++ function), 209
chan_list_init (C++ function), 209
chan_list_poll (C++ function), 210
chan_list_remove (C++ function), 209
chan_list_t (C++ class), 210
chan_list_t::max (C++ member), 211
chan_list_t::max (C++ member), 211
chan_module_init (C++ function), 207
chan_null (C++ function), 210
chan_poll (C++ function), 210
chan_read (C++ function), 208
chan_read_fn_t (C++ type), 206
chan_read_null (C++ function), 210
chan_set_control_cb (C++ function), 208
chan_set_write_filter_cb (C++ function), 207
chan_set_write_filter_isr_cb (C++ function), 207
chan_set_write_isr_cb (C++ function), 207
chan_size (C++ function), 208
chan_size_fn_t (C++ type), 206
chan_size_null (C++ function), 210
chan_t (C++ class), 211
chan_t::control (C++ member), 211
chan_t::list_p (C++ member), 211
chan_t::read (C++ member), 211
chan_t::reader_p (C++ member), 211
chan_t::write (C++ member), 211
chan_t::write_filter_cb (C++ member), 211
chan_t::write_filter_isr_cb (C++ member), 211
chan_t::write_isr (C++ member), 211
chan_write (C++ function), 208
chan_write_fn_t (C++ type), 206
chan_write_null (C++ function), 210
chipid (module), 226
chipid_read (C++ function), 226
circular_buffer (module), 391
circular_buffer_array_one (C++ function), 392
circular_buffer_array_two (C++ function), 392
<table>
<thead>
<tr>
<th>Configuration Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_FS_CMD_NVM_READ (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_NVM_WRITE (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_READ (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PIN_WRITE (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_PING_PING (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_LIST (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_START (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SERVICE_STOP (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_LIST (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_READ (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_RESET (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SETTINGS_WRITE (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_BACKTRACE (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_CONFIG (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_INFO (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_PANIC (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_REBOOT (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_RESET_CAUSE (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_SYS_UPTIME (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_THRD_LIST (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_DEVICE_LIST (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_CMD_USB_HOST_LIST (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_FS_PATH_MAX (C macro)</td>
<td>15</td>
</tr>
<tr>
<td>CONFIG_HARNESS_EXPECT_BUFFER_SIZE (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HARNESS_HEAP_MAX (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HARNESS_MOCK_VERBOSE (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HARNESS_SLEEP_MS (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_REQUEST_BUFFER_SIZE (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_HTTP_SERVER_SSL (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_I2C (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_I2C_SOFT (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_LED_7SEG_HT16K33 (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_LINUX_SOCKET_DEVICE (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_MCP2515 (C macro)</td>
<td>16</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_OUTPUT_PIN (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ANALOG_INPUT_PIN (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_BUS (C macro)</td>
<td>14</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CAN (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHAN (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_CHIPID (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DAC (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS18B20 (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_DS2321 (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_ESP_WIFI (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_EXTI (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FLASH (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_FS (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_I2C_SOFT (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_INET (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_IPV6 (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_KERN (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_LOG (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_MCP2515 (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NETWORK_INTERFACE (C macro)</td>
<td>14</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_NRF24L01 (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_OCI (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PIN (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PING (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_PWM_SOFT (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RANDOM (C macro)</td>
<td>13</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_RWLOCK (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SD (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SEM (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SETTINGS (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_SOCKET (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_THRD (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_TIMER (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UART_SOFT (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_UPGRADE (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_DEVICE (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_USB_HOST (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MODULE_INIT_WATCHDOG (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_MONITOR_THREAD_PERIOD_US (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_NRF24L01 (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_NVMEEPROM_SOFT (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_NVMEEPROM_SOFT_BLOCK_0_SIZE (C macro)</td>
<td>12</td>
</tr>
<tr>
<td>CONFIG_NVMEEPROM_SOFT_BLOCK_1_SIZE (C macro)</td>
<td>12</td>
</tr>
</tbody>
</table>
console (module), 362
counsel_get_input_channel (C++ function), 363
counsel_get_output_channel (C++ function), 363
counsel_init (C++ function), 363
counsel_module_init (C++ function), 363
counsel_set_input_channel (C++ function), 363
counsel_set_output_channel (C++ function), 363
counsel_start (C++ function), 363
counsel_stop (C++ function), 363
container_of (C macro), 201
cpu_usage_t (C++ type), 186
crc (module), 416
crc_32 (C++ function), 416
crc_7 (C++ function), 417
crc_8 (C++ function), 417
CRC_8_POLYNOMIAL_8_5_4_0 (C macro), 416
crc_ccitt (C++ function), 416
crc_xmodem (C++ function), 417
cst (C macro), 202
dac (module), 226
dac_0_dev (C macro), 426, 436, 440, 442
dac_async_convert (C++ function), 227
dac_async_wait (C++ function), 227
dac_convert (C++ function), 227
dac_device (C++ member), 227
DAC_DEVICE_MAX (C macro), 454, 455
dac_init (C++ function), 226
dac_module_init (C++ function), 226
date_t (C++ class), 302
date_t::attributes (C++ member), 303
date_t::creation_date (C++ member), 303
date_t::creation_time (C++ member), 303
date_t::creation_time_tenths (C++ member), 303
date_t::file_size (C++ member), 303
date_t::first_cluster_high (C++ member), 303
date_t::first_cluster_low (C++ member), 303
date_t::last_access_date (C++ member), 303
date_t::last_write_date (C++ member), 303
date_t::last_write_time (C++ member), 303
date_t::name (C++ member), 303
date_t::reserved1 (C++ member), 303
DIV_CEIL (C macro), 202
DIV_ROUND (C macro), 202
dor0 (C macro), 454
ds18b20 (module), 227
ds18b20_convert (C++ function), 228
ds18b20_driver_t (C++ class), 229
ds18b20_driver_t::next_p (C++ member), 229
ds18b20_driver_t::owi_p (C++ member), 229
dS18B20_FAMILY_CODE (C macro), 228
ds18b20_get_temperature (C++ function), 228
ds18b20_get_temperature_str (C++ function), 228
ds18b20_init (C++ function), 228
ds18b20_module_init (C++ function), 228
ds3231 (module), 229
ds3231_driver_t (C++ class), 229
ds3231_driver_t::i2c_p (C++ member), 230
ds3231_get_date (C++ function), 229
ds3231_init (C++ function), 229
ds3231_set_date (C++ function), 229

direction_t (C++ class), 302
direction_t::attributes (C++ member), 303
direction_t::creation_date (C++ member), 303
direction_t::creation_time (C++ member), 303
direction_t::creation_time_tenths (C++ member), 303
direction_t::file_size (C++ member), 303
direction_t::first_cluster_high (C++ member), 303
direction_t::first_cluster_low (C++ member), 303
direction_t::last_access_date (C++ member), 303
direction_t::last_write_date (C++ member), 303
direction_t::last_write_time (C++ member), 303
direction_t::name (C++ member), 303
direction_t::reserved1 (C++ member), 303

d呢 (C macro), 202
dIV_ROUND (C macro), 202
dor0 (C macro), 454
ds18b20 (module), 227
ds18b20_convert (C++ function), 228
ds18b20_driver_t (C++ class), 229
ds18b20_driver_t::next_p (C++ member), 229
ds18b20_driver_t::owi_p (C++ member), 229
dS18B20_FAMILY_CODE (C macro), 228
ds18b20_get_temperature (C++ function), 228
ds18b20_get_temperature_str (C++ function), 228
ds18b20_init (C++ function), 228
ds18b20_module_init (C++ function), 228
ds3231 (module), 229
ds3231_driver_t (C++ class), 229
ds3231_driver_t::i2c_p (C++ member), 230
ds3231_get_date (C++ function), 229
ds3231_init (C++ function), 229
ds3231_set_date (C++ function), 229

direction_t (C++ class), 302
direction_t::attributes (C++ member), 303
direction_t::creation_date (C++ member), 303
direction_t::creation_time (C++ member), 303
direction_t::creation_time_tenths (C++ member), 303
direction_t::file_size (C++ member), 303
direction_t::first_cluster_high (C++ member), 303
direction_t::first_cluster_low (C++ member), 303
direction_t::last_access_date (C++ member), 303
direction_t::last_write_date (C++ member), 303
direction_t::last_write_time (C++ member), 303
direction_t::name (C++ member), 303
direction_t::reserved1 (C++ member), 303

E2BIG (C macro), 179
EACCES (C macro), 180
EADDRINUSE (C macro), 184
EADDRNOTAVAIL (C macro), 184
EADV (C macro), 182
EAFNOSUP (C macro), 183
EBADMSG (C macro), 183
EBADR (C macro), 182
EBADRQC (C macro), 182
EBADSLT (C macro), 182
EBFONT (C macro), 182
EBTASSERT (C macro), 185
EBUSY (C macro), 180
ECANCELED (C macro), 185
ECANCELED (C macro), 185
ECANCELED (C macro), 185

ECHILD (C macro), 180
ECHRN (C macro), 181
ECOMM (C macro), 182
ECONNABORTED (C macro), 184
ECONNREFUSED (C macro), 184
ECONNRESET (C macro), 184
EDADLK (C macro), 181
EDADLOCK (C macro), 182
EDESTADDRREQ (C macro), 183
EDOM (C macro), 181
EDOTDOT (C macro), 183
EDIQUOT (C macro), 185
eeprom_soft (module), 230
eeprom_soft_block_t (C++ class), 231
eeprom_soft_block_t::address (C++ member), 231
eeprom_soft_block_t::size (C++ member), 231
eeprom_soft_driver_t (C++ class), 231
eeprom_soft_driver_t::block_p (C++ member), 231
eeprom_soft_driver_t::blocks_p (C++ member), 231
eeprom_soft_driver_t::chunk_address (C++ member), 231
eeprom_soft_driver_t::chunk_size (C++ member), 231
eeprom_soft_driver_t::eprom_size (C++ member), 231
eeprom_soft_driver_t::flash_p (C++ member), 231
eeprom_soft_driver_t::number_of_blocks (C++ member), 231
eeprom_soft_driver_t::revision (C++ member), 231
eeprom_soft_format (C++ function), 230
eeprom_soft_init (C++ function), 230
eeprom_soft_module_init (C++ function), 230
eeprom_soft_mount (C++ function), 230
eeprom_soft_read (C++ function), 230
eeprom_soft_write (C++ function), 231
EXIT (C macro), 180
EFAULT (C macro), 180
EFBIG (C macro), 180
EHOSTDOWN (C macro), 184
EHOSTUNREACH (C macro), 184
EIDRM (C macro), 181
EILSEQ (C macro), 183
EINPROGRESS (C macro), 185
EINTR (C macro), 179
EINVAL (C macro), 180
EIO (C macro), 180
EISCONN (C macro), 184
EISDIR (C macro), 180
EISNAM (C macro), 185
EKEYEXPIRED (C macro), 185
EKEYREJECTED (C macro), 185
EKEYREVOKED (C macro), 185
EL2HLT (C macro), 182
EL2NSYNC (C macro), 183
EL3HLT (C macro), 181
EL3RST (C macro), 181
ELIBACC (C macro), 183
ELIBBAD (C macro), 183
ELIBEXEC (C macro), 183
ELIBMAX (C macro), 183
ELIBSCN (C macro), 183
ELNRNG (C macro), 181
ELOOP (C macro), 181
emacs (C++ function), 405
emacs (module), 405
EMEDIUMTYPE (C macro), 185
EMFILE (C macro), 180
ENFILE (C macro), 180
ENOANO (C macro), 182
ENOBUFS (C macro), 184
ENOCOMMAND (C macro), 185
ENOCSI (C macro), 181
ENODATA (C macro), 182
ENODEV (C macro), 180
ENOENT (C macro), 179
ENOEXEC (C macro), 180
ENOFILE (C macro), 180
ENOLOCK (C macro), 181
ENOLINK (C macro), 181
ENOMEDIUM (C macro), 185
ENOMEM (C macro), 180
ENOMSG (C macro), 181
ENONET (C macro), 182
ENOPKG (C macro), 182
ENDPOINT_ATTRIBUTES_SYNCHRONISATION_TYPE (C macro), 275
ENDPOINT_ATTRIBUTES_TRANSFER_TYPE (C macro), 275
ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_BULK (C macro), 275
ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_CONTROL (C macro), 275
ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_INTERRUPT (C macro), 275
ENDPOINT_ATTRIBUTES_TRANSFER_TYPE_ISOCHRONOUS (C macro), 275
ENDPOINT_ATTRIBUTES_USAGE_TYPE (C macro), 275
ENDPOINT_ENDPOINT_ADDRESS_DIRECTION (C macro), 275
ENDPOINT_ENDPOINT_ADDRESS_NUMBER (C macro), 275
ENETDOWNGATEWAY (C macro), 184
ENETRESET (C macro), 184
ENETUNREACH (C macro), 184
ENFILE (C macro), 180
ENOANO (C macro), 182
ENOBUFS (C macro), 184
ENOCOMMAND (C macro), 185
ENOCSI (C macro), 181
ENODATA (C macro), 182
ENODEV (C macro), 180
ENOENT (C macro), 179
ENOEXEC (C macro), 180
ENOFILE (C macro), 180
ENOLOCK (C macro), 181
ENOLINK (C macro), 181
ENOMEDIUM (C macro), 185
ENOMEM (C macro), 180
ENOMSG (C macro), 181
ENONET (C macro), 182
ENOPKG (C macro), 182
Index 477
ENOPROTOOPT (C macro), 183
ENOSPC (C macro), 180
ENOSR (C macro), 182
ENOSTR (C macro), 182
ENOSYS (C macro), 181
ENOTBLK (C macro), 180
ENOTCONN (C macro), 184
ENOTDIR (C macro), 180
ENOTEMPTY (C macro), 181
ENOTNAM (C macro), 185
ENOTSOCK (C macro), 183
ENOTTY (C macro), 180
ENOTUNIQ (C macro), 183
ENXIO (C macro), 179
EOPNOTSUPP (C macro), 184
EOVERFLOW (C macro), 183
EPERM (C macro), 179
EPFNOSUPBOARD (C macro), 184
EPIPE (C macro), 181
EPROTO (C macro), 182
EPROTONOSUPBOARD (C macro), 184
EPROTOTYPE (C macro), 183
ERANGE (C macro), 181
EREMCHG (C macro), 183
EREMOTE (C macro), 182
EREMOTEIO (C macro), 185
ERESTART (C macro), 183
EROFS (C macro), 181
errno (module), 179
ESHUTDOWN (C macro), 184
ESOCKTNOSUPBOARD (C macro), 184
esp01 (module), 433
esp12e (module), 434
esp32_devkitc (module), 435
esp8266 (module), 455
esp_wifi (module), 232
esp_wifi_dhcp_status_running_t (C++ enumerator), 236
esp_wifi_dhcp_status_stopped_t (C++ enumerator), 236
esp_wifi_dhcp_status_t (C++ type), 236
esp_wifi_print (C++ function), 237
esp_wifi_set_op_mode (C++ function), 236
esp_wifi_set_phy_mode (C++ function), 236
esp_wifi_softap (module), 232
esp_wifi_softap_dhcp_server_start (C++ function), 233
esp_wifi_softap_dhcp_server_status (C++ function), 233
esp_wifi_softap_dhcp_server_stop (C++ function), 233
esp_wifi_get_ip_info (C++ function), 233
esp_wifi_get_number_of_connected_stations (C++ function), 233
esp_wifi_softap_set_station_info (C++ function), 233
esp_wifi_softap_init (C++ function), 232
esp_wifi_softap_set_ip_info (C++ function), 233
esp_wifi_station (module), 234
esp_wifi_station_connect (C++ function), 234
esp_wifi_station_dhcp_client_start (C++ function), 235
esp_wifi_station_dhcp_client_status (C++ function), 235
esp_wifi_station_disconnect (C++ function), 234
esp_wifi_station_get_ip_info (C++ function), 235
esp_wifi_station_get_reconnect_policy (C++ function), 235
esp_wifi_station_get_status (C++ function), 235
esp_wifi_station_init (C++ function), 234
esp_wifi_station_get_ip_info (C++ function), 234
esp_wifi_station_get_reconnect_policy (C++ function), 235
esp_wifi_station_status_as_string (C++ function), 235
esp_wifi_station_status_auth_failure_t (C++ enumerator), 234
esp_wifi_station_status_connected_t (C++ enumerator), 234
esp_wifi_station_status_connecting_t (C++ enumerator), 234
esp_wifi_station_status_connected_t (C++ enumerator), 234
esp_wifi_station_status_connecting_t (C++ enumerator), 234
esp_wifi_station_status_got_ip_t (C++ enumerator), 234
esp_wifi_station_status_id_t (C++ enumerator), 234
esp_wifi_station_status_no_ap_found_t (C++ enumerator), 234
esp_wifi_station_status_t (C++ type), 234
ESPIPE (C macro), 180
ESRCH (C macro), 179
ESRMNT (C macro), 182
ESTACK (C macro), 182
ESTALE (C macro), 185
ETRANS (C macro), 183
ETIME (C macro), 182
ETIMEDOUT (C macro), 184
flash_read (C++ function), 239
flash_write (C++ function), 239
fs (module), 305
FS_APPEND (C macro), 307
fs_auto_complete (C++ function), 311
fs_call (C++ function), 308
fs_callback_t (C++ type), 307
fs_close (C++ function), 309
fs_command_deregister (C++ function), 313
fs_command_init (C++ function), 312
fs_command_register (C++ function), 313
fs_command_t (C++ class), 315
fs_command_t::arg_p (C++ member), 315
fs_command_t::callback (C++ member), 315
fs_command_t::next_p (C++ member), 315
fs_command_t::path_p (C++ member), 315
fs_counter_deregister (C++ function), 313
fs_counter_increment (C++ function), 313
fs_counter_init (C++ function), 313
fs_counter_register (C++ function), 313
fs_counter_t (C++ class), 315
fs_counter_t::command (C++ member), 316
fs_counter_t::next_p (C++ member), 316
FS_CREAT (C macro), 307
fs_dir_close (C++ function), 310
fs_dir_entry_t (C++ class), 316
fs_dir_entry_t::latest_mod_date (C++ member), 316
fs_dir_entry_t::name (C++ member), 316
fs_dir_entry_t::size (C++ member), 316
fs_dir_entry_t::type (C++ member), 316
fs_dir_open (C++ function), 310
fs_dir_read (C++ function), 310
fs_dir_t (C++ class), 316
fs_dir_t::filesystem_p (C++ member), 316
FS_EXCL (C macro), 307
fs_file_t (C++ class), 315
fs_file_t::filesystem_p (C++ member), 315
FS_RDWR (C macro), 307
fs_filesystem_deregister (C++ function), 312
fs_filesystem_init_generic (C++ function), 312
fs_filesystem_operations_t (C++ class), 316
fs_filesystem_operations_t::file_close (C++ member), 316
fs_filesystem_operations_t::file_open (C++ member), 316
fs_filesystem_operations_t::file_read (C++ member), 316
fs_filesystem_operations_t::file_seek (C++ member), 316
fs_filesystem_operations_t::file_tell (C++ member), 316
fs_filesystem_operations_t::file_write (C++ member), 316
fs_filesystem_register (C++ function), 312
fs_filesystem_t::type (C++ member), 315
fs_format (C++ function), 311
fs_list (C++ function), 311
fs_module_init (C++ function), 308
fs_open (C++ function), 308
fs_parameter_deregister (C++ function), 314
fs_parameter_init (C++ function), 314
fs_parameter_int_print (C++ function), 314
fs_parameter_int_set (C++ function), 314
fs_parameter_print_callback_t (C++ type), 307
fs_parameter_register (C++ function), 314
fs_parameter_set_callback_t (C++ type), 307
fs_parameter_t (C++ class), 316
fs_parameter_t::command (C++ member), 316
fs_parameter_t::next_p (C++ member), 316
fs_parameter_t::print_cb (C++ member), 316
fs_parameter_t::set_cb (C++ member), 316
fs_parameter_t::value_p (C++ member), 316
FS_RDWR (C macro), 307
FS_READ (C macro), 306
fs_read (C++ function), 309
fs_read_line (C++ function), 309
fs_remove (C++ function), 310
fs_seek (C++ function), 309
FS_SEEK_CUR (C macro), 306
FS_SEEK_END (C macro), 306
FS_SEEK_SET (C macro), 306
fs_split (C++ function), 311
fs_stat (C++ function), 310
fs_stat_t (C++ class), 315
fs_stat_t::size (C++ member), 315
fs_stat_t::type (C++ member), 315
FS_SYNC (C macro), 307
fs_tell (C++ function), 310
FS_TRUNC (C macro), 307
FS_TYPE_DIR (C macro), 307
fs_type_fat16_t (C++ enumerator), 308
FS_TYPE_FILE (C macro), 307
fs_type_generic_t (C++ enumerator), 308
FS_TYPE_HARD_LINK (C macro), 307
FS_TYPE_SOFT_LINK (C macro), 307
fs_type_spiffs_t (C++ enumerator), 308
fs_type_t (C++ type), 308
FS_WRITE (C macro), 306
fs_write (C++ function), 309
harness (module), 380
harness_expect (C++ function), 383
harness_init (C++ function), 383
harness_mock_read (C++ function), 383
JSON_STRING (C++ enumerator), 412
json_t (C++ class), 415
json_t::num_tokens (C++ member), 416
json_t::tokens_p (C++ member), 416
json_t::toknext (C++ member), 416
json_t::toksuper (C++ member), 416
json_tok_t (C++ class), 415
json_tok_t::buf_p (C++ member), 415
json_tok_t::num_tokens (C++ member), 415
json_tok_t::size (C++ member), 415
json_tok_t::type (C++ member), 415
json_token_array (C++ function), 414
json_token_false (C++ function), 415
json_token_null (C++ function), 415
json_token_number (C++ function), 415
json_token_object (C++ function), 414
json_token_string (C++ function), 415
json_token_true (C++ function), 414
json_type_t (C++ type), 412
JSON_UNDEFINED (C++ enumerator), 412
L
led_7seg_ht16k33 (module), 244
led_7seg_ht16k33_brightness (C++ function), 246
LED_7SEG_H16K33_BRIGHTNESS_MAX (C macro), 245
LED_7SEG_H16K33_BRIGHTNESS_MIN (C macro), 245
led_7seg_ht16k33_clear (C++ function), 245
LED_7SEG_H16K33_DEFAULT_I2C_ADDR (C macro), 245
led_7seg_ht16k33_display (C++ function), 245
led_7seg_ht16k33_driver_t (C++ class), 246
led_7seg_ht16k33_driver_t::buf (C++ member), 247
led_7seg_ht16k33_driver_t::i2c_addr (C++ member), 247
led_7seg_ht16k33_driver_t::i2c_p (C++ member), 247
led_7seg_ht16k33_init (C++ function), 245
led_7seg_ht16k33_module_init (C++ function), 245
led_7seg_ht16k33_set_num (C++ function), 246
led_7seg_ht16k33_show_colon (C++ function), 246
led_7seg_ht16k33_show_dot (C++ function), 246
led_7seg_ht16k33_start (C++ function), 245
linux (module), 438, 455
list (module), 396
list_next_t (C++ class), 397
list_next_t::next_p (C++ member), 398
list singly_linked_t (C++ class), 398
list singly_linked_t::head_p (C++ member), 398
list singly_linked_t::tail_p (C++ member), 398
LIST_SL_ADD_HEAD (C macro), 396
LIST_SL_ADD_TAIL (C macro), 397
LIST_SL_INIT (C macro), 396
LIST_SL_INIT_STRUCT (C macro), 396
LIST_SL_ITERATOR_INIT (C macro), 397
LIST_SL_ITERATOR_NEXT (C macro), 397
LOG_ALL (C macro), 386
LOG_DEBUG (C macro), 386
LOG_ERROR (C macro), 386
LOG_FATAL (C macro), 386
LOG_NONE (C macro), 386
LOG_WARNING (C macro), 386
M
maple_esp32 (module), 439
MAX (C macro), 202
mbr_t (C++ class), 302
mbr_t::codeArea (C++ member), 302
mbr_t::diskSignature (C++ member), 302
mbr_t::mbr_sig (C++ member), 302
mcp2515 (module), 247
mcp2515_driver_t (C++ class), 248
mcp2515_driver_t::chin_p (C++ member), 249
mcp2515_driver_t::chout (C++ member), 249
MIDI_PERC_LOW_FLOOR_TOM (C macro), 421
MIDI_PERC_LOW_MID_TOM (C macro), 421
MIDI_PERC_LOW_TIMBALE (C macro), 421
MIDI_PERC_LOW_TOM (C macro), 421
MIDI_PERC_LOW_WOOD_BLOCK (C macro), 422
MIDI_PERC_MARACAS (C macro), 421
MIDI_PERC_MUTE_CUICA (C macro), 422
MIDI_PERC_MUTE_HI_CONGA (C macro), 421
MIDI_PERC_MUTE_TRIANGLE (C macro), 422
MIDI_PERC_OPEN_CUICA (C macro), 422
MIDI_PERC_OPEN_HI_CONGA (C macro), 421
MIDI_PERC_OPEN_HI_HAT (C macro), 421
MIDI_PERC_OPEN_TRIANGLE (C macro), 422
MIDI_PERC_PEDAL_HI_HAT (C macro), 421
MIDI_PERC_RIDE_BELL (C macro), 421
MIDI_PERC_RIDE_CYMBAL_1 (C macro), 421
MIDI_PERC_RIDE_CYMBAL_2 (C macro), 421
MIDI_PERC_SHORT_GUIRO (C macro), 422
MIDI_PERC_SHORT_WHISTLE (C macro), 421
MIDI_PERC_SIDE_STICK (C macro), 421
MIDI_PERC_SPLASH_CYMBAL (C macro), 421
MIDI_PERC_TAMBOURINE (C macro), 421
MIDI_PERC_VIBRASLAP (C macro), 421
MIDI_PITCH_BEND_CHANGE (C macro), 419
MIDI_POLYPHONIC_KEY_PRESSURE (C macro), 419
MIN (C macro), 202
mqtt_application_message_t (C++ class), 344
mqtt_application_message_t::buf_p (C++ member), 344
mqtt_application_message_t::qos (C++ member), 344
mqtt_application_message_t::size (C++ member), 344
mqtt_client (module), 340
mqtt_client_connect (C++ function), 342
mqtt_client_disconnect (C++ function), 342
mqtt_client_init (C++ function), 342
mqtt_client_ping (C++ function), 343
mqtt_client::publish (C++ function), 343
mqtt_client_state_connected_t (C++ enumerator), 341
mqtt_client_state_connecting_t (C++ enumerator), 341
mqtt_client_state_disconnected_t (C++ enumerator), 341
mqtt_client_state_t (C++ type), 341
mqtt_client_subscribe (C++ function), 343
mqtt_client_t (C++ class), 344
mqtt_client_t::buf_p (C++ member), 344
mqtt_client_t::state (C++ member), 343
mqtt_client_t::type (C++ member), 344
mqtt_client.unsubscribe (C++ function), 343
mqtt.on_error (C++ type), 341
mqtt.on_publish_t (C++ type), 341
mqtt_qos_0_t (C++ enumerator), 342
mqtt_qos_1_t (C++ enumerator), 342
mqtt_qos_2_t (C++ enumerator), 342
mqtt_qos_t (C++ type), 341
N
nano32 (module), 441
network_interface (module), 344
network_interface_add (C++ function), 349
network_interface_driver_ESP (module), 346
network_interface_get_by_name (C++ function), 350
network_interface_get_ip_info (C++ function), 350
network_interface_get_ip_info_t (C++ type), 349
network_interface_is_up (C++ function), 349
network_interface_is_up_t (C++ type), 349
network_interface_module_init (C++ function), 349
network_interface_set_ip_info (C++ function), 350
network_interface_set_ip_info_t (C++ type), 349
network_interface_slip (module), 344
network_interface_slip_init (C++ function), 345
network_interface_slip_input (C++ function), 345
network_interface_slip_module_init (C++ function), 345
network_interface_slip_STATE_ESCAPE (C++ enumerator), 345
network_interface_slip_STATE_NORMAL (C++ enumerator), 345
network_interface_slip_state_t (C++ type), 345
network_interface_slip_t (C++ class), 346
network_interface_slip_t::buf_p (C++ member), 346
network_interface_slip_t::chout_p (C++ member), 346
network_interface_slip_t::network_interface (C++ member), 346
network_interface_slip_t::pbuf_p (C++ member), 346
network_interface_slip_t::size (C++ member), 346
network_interface_slip_t::state (C++ member), 346
network_interface_slip_t::type (C++ member), 346
network_interface_start (C++ function), 346
network_interface_start_t (C++ type), 346
network_interface_stop (C++ function), 346
network_interface_stop_t (C++ type), 346
network_interface_t (C++ class), 350
network_interface_t::get_ip_info (C++ member), 350
network_interface_t::info (C++ member), 350
network_interface_t::is_up (C++ member), 350
network_interface_t::name_p (C++ member), 350
network_interface_t::netif_p (C++ member), 350
network_interface_t::next_p (C++ member), 350
network_interface_t::set_ip_info (C++ member), 350
network_interface_t::start (C++ member), 350
network_interface_t::stop (C++ member), 350
network_interface_wifi (module), 346
network_interface_wifi_driver_esp_softap (C++ member), 346
network_interface_wifi_driver_esp_station (C++ member), 346
network_interface_wifi_driver_t (C++ class), 348
network_interface_wifi_driver_t::get_ip_info (C++ member), 348
network_interface_wifi_driver_t::init (C++ member), 348
network_interface_wifi_driver_t::is_up (C++ member), 348
network_interface_wifi_driver_t::set_ip_info (C++ member), 348
network_interface_wifi_driver_t::start (C++ member), 348
network_interface_wifi_driver_t::stop (C++ member), 348
network_interface_wifi_get_ip_info (C++ function), 347
network_interface_wifi_init (C++ function), 346
network_interface_wifi_is_up (C++ function), 347
network_interface_wifi_module_init (C++ function), 346
network_interface_wifi_set_ip_info (C++ function), 347
network_interface_wifi_start (C++ function), 347
network_interface_wifi_stop (C++ function), 347
network_interface_wifi_t (C++ class), 348
network_interface_wifi_t::arg_p (C++ member), 348
network_interface_wifi_t::driver_p (C++ member), 348
network_interface_wifi_t::info_p (C++ member), 348
network_interface_wifi_t::network_interface (C++ member), 348
network_interface_wifi_t::password_p (C++ member), 348
network_interface_wifi_t::ssid_p (C++ member), 348
nodemcu (module), 442
nrf24l01 (module), 249
nrf24l01_driver_t (C++ class), 250
nrf24l01_driver_t::address (C++ member), 250
nrf24l01_driver_t::ce (C++ member), 250
nrf24l01_driver_t::chin (C++ member), 250
nrf24l01_driver_t::chinbuf (C++ member), 250
nrf24l01_driver_t::exti (C++ member), 250
nrf24l01_driver_t::irqbuf (C++ member), 250
nrf24l01_driver_t::irqchan (C++ member), 250
nrf24l01_driver_t::spi (C++ member), 250
nrf24l01_driver_t::stack (C++ member), 250
nrf24l01_driver_t::thrd (C++ member), 250
nrf24l01_init (C++ function), 249
nrf24l01_module_init (C++ function), 249
nrf24l01_read (C++ function), 250
nrf24l01_start (C++ function), 249
nrf24l01_stop (C++ function), 249
nrf24l01_write (C++ function), 250
nvm (module), 363
nvm_format (C++ function), 364
nvm_module_init (C++ function), 364
nvm_mount (C++ function), 364
nvm_read (C++ function), 364
nvm_write (C++ function), 364

O
O_APPEND (C macro), 295
O_CREAT (C macro), 295
O_EXCL (C macro), 295
O_RDWR (C macro), 295
O_RDONLY (C macro), 295
O_RDWR (C macro), 295
O_TRUNC (C macro), 295
O_WRITE (C macro), 295
O_WRONLY (C macro), 295
OSTR (C macro), 202
owi (module), 251
OWI_ALARM_SEARCH (C macro), 251
owi_device_t (C++ class), 252
owi_device_t::id (C++ member), 252
owi_driver_t (C++ class), 252
owi_driver_t::devices_p (C++ member), 252
owi_driver_t::len (C++ member), 252
owi_driver_t::nmemb (C++ member), 252
owi_driver_t::pin (C++ member), 252
owi_init (C++ function), 251
OWI_MATCH_ROM (C macro), 251
owi_read (C++ function), 251
OWI_READ_ROM (C macro), 251
owi_reset (C++ function), 251
owi_search (C++ function), 251
OWI_SEARCH_ROM (C macro), 251
OWI_SKIP_ROM (C macro), 251
owi_write (C++ function), 252

P
PACKED (C++ member), 262, 299
PANIC_ASSERT (C macro), 179
PANIC_ASSERTN (C macro), 179
part_t (C++ class), 300
part_t::begin_cylinder_high (C++ member), 300
part_t::begin_cylinder_low (C++ member), 300
part_t::begin_head (C++ member), 300
part_t::begin_sector (C++ member), 300
part_t::boot (C++ member), 300
part_t::end_cylinder_high (C++ member), 300
part_t::end_cylinder_low (C++ member), 300
part_t::end_head (C++ member), 300
part_t::end_sector (C++ member), 300
part_t::first_sector (C++ member), 300

Index 487
pin_pe5_dev (C macro), 448
pin_pe6_dev (C macro), 448
pin_pe7_dev (C macro), 448
pin_pe8_dev (C macro), 448
pin_pe9_dev (C macro), 448
pin_read (C++ function), 253
pin_set_mode (C++ function), 254
pin_toggle (C++ function), 254
pin_write (C++ function), 253
ping (module), 351
ping_host_by_ip_address (C++ function), 351
ping_module_init (C++ function), 351
PRINT_FILE_LINE (C macro), 201
pwm (module), 255
pwm_a4_dev (C macro), 444
pwm_a5_dev (C macro), 444
pwm_d0_dev (C macro), 444
pwm_d10_dev (C macro), 426, 429–431, 433, 438
pwm_d11_dev (C macro), 426, 429–431, 433, 438
pwm_d12_dev (C macro), 426, 429
pwm_d1_dev (C macro), 444
pwm_d2_dev (C macro), 426, 429, 444
pwm_d3_dev (C macro), 426, 429–431, 433, 438, 444
pwm_d5_dev (C macro), 426, 429
pwm_d6_dev (C macro), 426, 429
pwm_d7_dev (C macro), 426, 429
pwm_d8_dev (C macro), 426, 429
pwm_d9_dev (C macro), 426, 429–431, 433, 438
pwm_device (C++ member), 257
PWM_DEVICE_MAX (C macro), 452, 453, 455
pwm_duty_cycle (C++ function), 257
pwm_duty_cycle_as_percent (C++ function), 257
pwm_frequency (C++ function), 256
pwm_frequency_as_hertz (C++ function), 256
pwm_get_duty_cycle (C++ function), 256
pwm_get_frequency (C++ function), 256
pwm_init (C++ function), 255
pwm_module_init (C++ function), 255
pwm_pin_to_device (C++ function), 257
pwm_set_duty_cycle (C++ function), 256
pwm_set_frequency (C++ function), 256
pwm_soft (module), 257
pwm_soft_driver_t (C++ class), 259
pwm_soft_driver_t::delta (C++ member), 260
pwm_soft_driver_t::duty_cycle (C++ member), 260
pwm_soft_driver_t::frequency (C++ member), 260
pwm_soft_driver_t::next_p (C++ member), 260
pwm_soft_driver_t::pin_dev_p (C++ member), 260
pwm_soft_driver_t::thrd_p (C++ member), 260
pwm_soft_duty_cycle (C++ function), 259
pwm_soft_duty_cycle_as_percent (C++ function), 259
pwm_soft_get_duty_cycle (C++ function), 259
pwm_soft_get_frequency (C++ function), 258
pwm_soft_init (C++ function), 258
pwm_soft_module_init (C++ function), 258
pwm_soft_set_duty_cycle (C++ function), 259
pwm_soft_set_frequency (C++ function), 258
pwm_soft_start (C++ function), 258
pwm_soft_stop (C++ function), 259
pwm_start (C++ function), 255
pwm_stop (C++ function), 255
Q
queue (module), 213
queue_buffer_t (C++ class), 215
queue_buffer_t::begin_p (C++ member), 215
queue_buffer_t::end_p (C++ member), 215
queue_buffer_t::read_p (C++ member), 215
queue_buffer_t::size (C++ member), 215
queue_buffer_t::write_p (C++ member), 215
queue_init (C++ function), 214
QUEUE_INITDECL (C macro), 213
queue_read (C++ function), 214
queue_size (C++ function), 215
queue_start (C++ function), 214
QUEUE_STATE_INITIALIZED (C++ enumerator), 213
QUEUE_STATE_RUNNING (C++ enumerator), 213
QUEUE_STATE_STOPPED (C++ enumerator), 214
queue_state_t (C++ type), 213
queue_stop (C++ function), 214
queue_stop_isr (C++ function), 214
queue_t (C++ class), 216
queue::base (C++ member), 216
queue::buf_p (C++ member), 216
queue::buffer (C++ member), 216
queue::left (C++ member), 216
queue::size (C++ member), 216
queue::state (C++ member), 216
queue::writer_p (C++ member), 216
queue::writers (C++ member), 216
queue_unused_size (C++ function), 215
queue_unused_size_isr (C++ function), 215
queue_write (C++ function), 214
queue_write_isr (C++ function), 215
R
random (module), 260
random_module_init (C++ function), 260
random_read (C++ function), 260
re (module), 405
re_compile (C++ function), 406
RE_DOTALL (C macro), 406
re_group_t (C++ class), 407
re_group_t::buf_p (C++ member), 407
re_group_t::size (C++ member), 407
RE_IGNORECASE (C macro), 406
re_match (C++ function), 407
RE_MULTILINE (C macro), 406
REQUEST_GET_DESCRIPTOR (C macro), 274
REQUEST_GET_STATUS (C macro), 274
REQUEST_SET_ADDRESS (C macro), 274
REQUEST_SET_CONFIGURATION (C macro), 274
REQUEST_TYPE_DATA_DIRECTION_DEVICE_TO_HOST (C macro), 274
REQUEST_TYPE_DATA_DIRECTION_HOSTToDevice (C macro), 274
REQUEST_TYPE_DATA_MASK (C macro), 274
REQUEST_TYPE_RECIPIENT_DEVICE (C macro), 274
REQUEST_TYPE_RECIPIENT_ENDPOINT (C macro), 274
REQUEST_TYPE_RECIPIENT_INTERFACE (C macro), 274
REQUEST_TYPE_RECIPIENT_MASK (C macro), 274
REQUEST_TYPE_RECIPIENT_OTHER (C macro), 274
REQUEST_TYPE_TYPE_CLASS (C macro), 274
REQUEST_TYPE_TYPE_MASK (C macro), 274
REQUEST_TYPE_TYPE_STANDARD (C macro), 274
REQUEST_TYPE_TYPE_VENDOR (C macro), 274
rwlock (module), 216
rwlock_init (C++ function), 217
rwlock_module_init (C++ function), 217
rwlock_reader_give (C++ function), 217
rwlock_reader_give_isr (C++ function), 217
rwlock_reader_take (C++ function), 217
rwlock_t (C++ class), 218
rwlock_t::number_of_readers (C++ member), 218
rwlock_t::number_of_writers (C++ member), 218
rwlock_t::readers_p (C++ member), 218
rwlock_t::writers_p (C++ member), 218
rwlock_writer_give (C++ function), 217
rwlock_writer_give_isr (C++ function), 218
rwlock_writer_take (C++ function), 217
RXCIE0 (C macro), 454
RXEN0 (C macro), 454
s16_t (C++ type), 202
s32_t (C++ type), 202
s8_t (C++ type), 202
sam3x8e (module), 455
SAM_PA (C macro), 456
SAM_PB (C macro), 456
SAM_PC (C macro), 456
SAM_PD (C macro), 456
sd (module), 260
SD_BLOCK_SIZE (C macro), 261
SD_C_SIZE (C macro), 261
SD_C_SIZE_MULT (C macro), 261
SD_CCC (C macro), 261
sd_cid_t (C++ class), 262
sd_cid_t::crc (C++ member), 262
sd_cid_t::mdt (C++ member), 262
sd_cid_t::mid (C++ member), 262
sd_cid_t::oid (C++ member), 262
sd_cid_t::pnm (C++ member), 262
sd_cid_t::prv (C++ member), 262
SD_CSD_STRUCTURE_V1 (C macro), 261
SD_CSD_STRUCTURE_V2 (C macro), 261
sd_csdt (C++ type), 265
sd_csdt::v1 (C++ member), 265
sd_csdt::v2 (C++ member), 265
sd_csdt_t (C++ class), 263
sd_csdt_t::c_size_high (C++ member), 263
sd_csdt_t::c_size_low (C++ member), 263
sd_csdt_t::c_size_mid (C++ member), 263
sd_csdt_t::c_size_mult_high (C++ member), 263
sd_csdt_t::c_size_mult_low (C++ member), 263
sd_csdt_t::ccc_high (C++ member), 263
sd_csdt_t::ccc_low (C++ member), 263
sd_csdt_t::copy (C++ member), 264
sd_csdt_t::crc (C++ member), 264
sd_csdt_t::csd_structure (C++ member), 263
sd_csdt_t::dsr_imp (C++ member), 263
sd_csdt_t::erase_blk_en (C++ member), 263
sd_csdt_t::file_format (C++ member), 264
sd_csdt_t::file_format_grp (C++ member), 264
sd_csdt_t::nsac (C++ member), 263
sd_csdt_t::perm_write_protect (C++ member), 264
sd_csdt_t::r2w_factor (C++ member), 263
sd_csdt_t::read_bl_len (C++ member), 263
sd_csdt_t::read_bl_partial (C++ member), 263
sd_csdt_t::read_blk_misalign (C++ member), 263
sd_csdt_t::reserved1 (C++ member), 263
sd_csdt_t::reserved2 (C++ member), 263
sd_csdt_t::reserved3 (C++ member), 263
sd_csdt_t::reserved4 (C++ member), 263
sd_csdt_t::reserved5 (C++ member), 263
sd_csdt_t::sector_size_high (C++ member), 263
sd_csdt_t::sector_size_low (C++ member), 263
sd_csdt_t::taac (C++ member), 263
sd_csdt_t::tmp_write_protect (C++ member), 264
sd_csdt_t::vdd_r_curr_max (C++ member), 263
sd_csdt_t::vdd_r_curr_min (C++ member), 263
sd_csdt_t::vdd_w_curr_max (C++ member), 263
sd_csdt_t::vdd_w_curr_min (C++ member), 263
sd_csdt_t::wp_grp_enable (C++ member), 263
sd_csdt_t::wp_grp_size (C++ member), 263
SD_CSD_STRUCTURE_V1 (C macro), 261
SD_CSD_STRUCTURE_V2 (C macro), 261
Index 491
setting_type_t (C++ type), 369
settings (module), 367
SETTINGS_AREA_CRC_OFFSET (C macro), 368
settings_module_init (C++ function), 369
settings_read (C++ function), 369
settings_read_by_name (C++ function), 369
settings_reset (C++ function), 370
settings_write (C++ function), 369
settings_write_by_name (C++ function), 369
sha1 (module), 417
sha1_digest (C++ function), 418
sha1_init (C++ function), 418
sha1_t (C++ class), 418
sha1_t::buf (C++ member), 418
sha1_t::h (C++ member), 418
sha1_t::size (C++ member), 418
sha1_update (C++ function), 418
shell (module), 370
shell_history_elem_t (C++ class), 371
shell_history_elem_t::buf (C++ member), 372
shell_history_elem_t::next_p (C++ member), 372
shell_history_elem_t::prev_p (C++ member), 372
shell_init (C++ function), 371
shell_line_t (C++ class), 372
shell_line_t::buf (C++ member), 372
shell_line_t::cursor (C++ member), 372
shell_line_t::length (C++ member), 372
shell_main (C++ function), 371
shell_module_init (C++ function), 371
shell_t (C++ class), 372
shell_t::arg_p (C++ member), 372
shell_t::authorized (C++ member), 372
shell_t::buf (C++ member), 372
shell_t::carriage_return_received (C++ member), 372
shell_t::chin_p (C++ member), 372
shell_t::chout_p (C++ member), 372
shell_t::current_p (C++ member), 372
shell_t::head_p (C++ member), 372
shell_t::heap (C++ member), 372
shell_t::line (C++ member), 372
shell_t::line_valid (C++ member), 372
shell_t::match (C++ member), 372
shell_t::name_p (C++ member), 372
shell_t::newline_received (C++ member), 372
shell_t::password_p (C++ member), 372
shell_t::pattern (C++ member), 372
shell_t::prev_line (C++ member), 372
shell_t::tail_p (C++ member), 372
shell_t::username_p (C++ member), 372
SHT3X_DIS_I2C_ADDR_A (C macro), 266
SHT3X_DIS_I2C_ADDR_B (C macro), 266
sht3xd (module), 265
sht3xd_driver_t::i2c_addr (C++ member), 267
sht3xd_driver_t::i2c_p (C++ member), 267
sht3xd_driver_t::serial (C++ member), 267
sht3xd_get_serial (C++ function), 267
sht3xd_get_temp_humid (C++ function), 266
sht3xd_init (C++ function), 266
sht3xd_module_init (C++ function), 266
sht3xd_start (C++ function), 266
soam (module), 373
soam_get_log_input_channel (C++ function), 375
soam_get_stdout_input_channel (C++ function), 375
soam_init (C++ function), 374
soam_input (C++ function), 374
soam_module_init (C++ function), 374
soam_t (C++ class), 376
soam_t::buf_p (C++ member), 376
soam_t::chout_p (C++ member), 376
soam_t::command_chan (C++ member), 376
soam_t::is_printf (C++ member), 376
soam_t::log_chan (C++ member), 376
soam_t::packet_index (C++ member), 376
soam_t::pos (C++ member), 376
soam_t::sem (C++ member), 376
soam_t::size (C++ member), 376
soam_t::stdout_chan (C++ member), 376
socket (module), 351
socket_accept (C++ function), 354
socket_bind (C++ function), 354
socket_close (C++ function), 353
socket_connect (C++ function), 354
socket_connect_by_hostname (C++ function), 354
SOCKET_DOMAIN_INET (C macro), 352
socket_listen (C++ function), 354
socket_module_init (C++ function), 353
socket_open (C++ function), 353
socket_open_raw (C++ function), 353
socket_open_tcp (C++ function), 353
socket_open_udp (C++ function), 353
SOCKET_PROTO_ICMP (C macro), 353
socket_read (C++ function), 355
socket_recvfrom (C++ function), 355
socket_sendto (C++ function), 355
socket_size (C++ function), 356
socket_t (C++ class), 356
socket_t::args_p (C++ member), 356
socket_t::base (C++ member), 356
socket_t::closed (C++ member), 356
socket_t::left (C++ member), 356
socket_t::dbuf_p (C++ member), 356
socket_t::dbuf_p (C++ member), 356
socket_t::pcb_p (C++ member), 356
Index 493
SPIFFS_ERR_FILE_EXISTS (C macro), 318
SPIFFS_ERR_FULL (C macro), 317
SPIFFS_ERR_INDEX_FREE (C macro), 318
SPIFFS_ERR_INDEX_INVALID (C macro), 318
SPIFFS_ERR_INDEX_LU (C macro), 318
SPIFFS_ERR_INDEX_REF_FREE (C macro), 318
SPIFFS_ERR_INDEX_REF_INVALID (C macro), 318
SPIFFS_ERR_INDEX_REF_LU (C macro), 318
SPIFFS_ERR_INDEX_SPAN_MISMATCH (C macro), 317
SPIFFS_ERR_INTERNAL (C macro), 318
SPIFFS_ERR_IS_FREE (C macro), 317
SPIFFS_ERR_IS_INDEX (C macro), 317
SPIFFS_ERR_MAGIC_NOT_POSSIBLE (C macro), 318
SPIFFS_ERR_MOUNTED (C macro), 318
SPIFFS_ERR_NAME_TOO_LONG (C macro), 318
SPIFFS_ERR_NO_DELETED_BLOCKS (C macro), 318
SPIFFS_ERR_NOT_A_FILE (C macro), 318
SPIFFS_ERR_NOT_A_FS (C macro), 318
SPIFFS_ERR_NOT_CONFIGURED (C macro), 318
SPIFFS_ERR_NOT_FINALIZED (C macro), 318
SPIFFS_ERR_NOT_INDEX (C macro), 318
SPIFFS_ERR_NOT_MOUNTED (C macro), 318
SPIFFS_ERR_NOT_READABLE (C macro), 318
SPIFFS_ERR_NOT_WRITABLE (C macro), 318
SPIFFS_ERR_OUT_OF_FILE_DESCS (C macro), 317
SPIFFS_ERR_PROBE_NOT_A_FS (C macro), 318
SPIFFS_ERR_PROBE_TOO_FEW_BLOCKS (C macro), 318
SPIFFS_ERR_RO_ABORTED_OPERATION (C macro), 318
SPIFFS_ERR_RO_NOT_IMPL (C macro), 318
SPIFFS_ERR_TEST (C macro), 318
spiffs_errno (C++ function), 325
spiffs_lseek (C++ function), 323
spiffs_mode (C++ type), 320
spiffs_mode_t (C++ type), 319
spiffs_mount (C++ function), 321
spiffs_mounted (C++ function), 326
SPIFFS_O_APPEND (C macro), 318
SPIFFS_O_CREAT (C macro), 318
SPIFFS_O_DIRECT (C macro), 319
SPIFFS_O_EXCL (C macro), 319
SPIFFS_O_RDONLY (C macro), 319
SPIFFS_O_RDWR (C macro), 319
SPIFFS_OK (C macro), 317
spiffs_open (C++ function), 322
spiffs_open_by_dirent (C++ function), 322
spiffs_open_by_page (C++ function), 322
spiffs_opendir (C++ function), 325
spiffs_page_ix (C++ type), 320
SPIFFS_RDONLY (C macro), 318
SPIFFS_RDWR (C macro), 319
spiffs_read (C++ function), 322
spiffs_read_cb_t (C++ type), 319
spiffs_readdir (C++ function), 325
spiffs_remove (C++ function), 323
spiffs_rename (C++ function), 324
SPIFFS_SEEK_CUR (C macro), 319
SPIFFS_SEEK_END (C macro), 319
SPIFFS_SEECK_SET (C macro), 319
spiffs_set_file_callback_func (C++ function), 327
spiffs_stat (C++ function), 324
spiffs_stat_t (C++ class), 329
spiffs_stat_t::name (C++ member), 329
spiffs_stat_t::obj_id (C++ member), 329
spiffs_stat_t::pix (C++ member), 329
spiffs_stat_t::size (C++ member), 329
spiffs_stat_t::type (C++ member), 329
spiffs_t (C++ class), 328
spiffs_t::block_count (C++ member), 328
spiffs_t::cfg (C++ member), 328
spiffs_t::check_cb_f (C++ member), 329
spiffs_t::cleaning (C++ member), 329
spiffs_t::config_magic (C++ member), 329
spiffs_t::cursor_block_i (C++ member), 328
spiffs_t::cursor_obj_lu_entry (C++ member), 328
spiffs_t::err_code (C++ member), 329
spiffs_t::fd_count (C++ member), 329
spiffs_t::fd_space (C++ member), 328
spiffs_t::file_cb_f (C++ member), 329
spiffs_t::format (C++ function), 326
spiffs_fremove (C++ function), 323
spiffs_fstat (C++ function), 324
spiffs_gc (C++ function), 327
SPIFFS_GC_DBG (C macro), 318
SPIFFS_GC_DBG (C macro), 318
SPIFFS_GC_DBG (C macro), 318
SPIFFS_GC_QUICK (C++ function), 326
spiiffs_info (C++ function), 326
SPIFFS_LOCK (C macro), 319
spiffs_lseek (C++ function), 323
spiffs_mode (C++ type), 320
spiffs_mode_t (C++ type), 319
spiffs_mount (C++ function), 321
spiffs_mounted (C++ function), 326
SPIFFS_O_APPEND (C macro), 318
SPIFFS_O_CREAT (C macro), 318
SPIFFS_O_DIRECT (C macro), 319
SPIFFS_O_EXCL (C macro), 319
SPIFFS_O_RDONLY (C macro), 319
SPIFFS_O_RDWR (C macro), 319
SPIFFS_OK (C macro), 317
sys_start (C++ function), 187
sys_stop (C++ function), 187
sys_t (C++ class), 189
sys_t::on_fatal_callback (C++ member), 190
sys_t::start (C++ member), 190
sys_t::stdin_p (C++ member), 190
sys_t::stdout_p (C++ member), 190
SYS_TICK_MAX (C macro), 186
sys_tick_t (C++ type), 186
sys_unlock (C++ function), 189
sys_unlock_isr (C++ function), 189
sys_uptime (C++ function), 187
sys_uptime_isr (C++ function), 188

T
t2st (C++ function), 186
tftp_server (module), 361
tftp_server_init (C++ function), 361
tftp_server_start (C++ function), 362
tftp_server_t (C++ class), 362
tftp_server_t::addr (C++ member), 362
tftp_server_t::listener (C++ member), 362
tftp_server_t::name_p (C++ member), 362
tftp_server_t::root_p (C++ member), 362
tftp_server_t::stack_p (C++ member), 362
tftp_server_t::stack_size (C++ member), 362
tftp_server_t::thrd_p (C++ member), 362
tftp_server_t::timeout_ms (C++ member), 362
thrd (module), 190
THRD_CONTEXT_LOAD_ISR (C macro), 191
THRD_CONTEXT_STORE_ISR (C macro), 191
thrd_environment_t (C++ class), 196
thrd_environment_t::max_number_of_variables (C++ member), 197
thrd_environment_t::number_of_variables (C++ member), 197
thrd_environment_t::variables_p (C++ member), 197
thrd_environment_variable_t (C++ class), 196
thrd_environment_variable_t::name_p (C++ member), 196
thrd_environment_variable_t::value_p (C++ member), 196
thrd_get_bottom_of_stack (C++ function), 195
thrd_get_by_name (C++ function), 193
thrd_get_env (C++ function), 195
thrd_get_global_env (C++ function), 194
thrd_get_log_mask (C++ function), 197
thrd_get_name (C++ function), 193
thrd_get_prio (C++ function), 197
thrd_get_top_of_stack (C++ function), 196
thrd_init_env (C++ function), 194
thrd_init_global_env (C++ function), 194
thrd_join (C++ function), 192
thrd_module_init (C++ function), 191
thrd_prio_list_elem_t (C++ class), 202
thrd_prio_list_elem_t::next_p (C++ member), 202
thrd_prio_list_init (C++ function), 196
thrd_prio_list_pop_isr (C++ function), 196
thrd_prio_list_push_isr (C++ function), 196
thrd_prio_list_remove_isr (C++ function), 196
thrd_prio_list_t (C++ class), 202
thrd_prio_list_t::head_p (C++ member), 202
THRD_RESCHEDULE_ISR (C macro), 191
thrd_resume (C++ function), 192
thrd_resume_isr (C++ function), 195
thrd_self (C++ function), 193
thrd_set_env (C++ function), 194
thrd_set_global_env (C++ function), 194
thrd_set_log_mask (C++ function), 193
thrd_set_name (C++ function), 193
thrd_set_prio (C++ function), 193
thrd_sleep (C++ function), 192
thrd_sleep_ms (C++ function), 192
thrd_sleep_us (C++ function), 193
thrd_spawn (C++ function), 191
THRD_STACK (C macro), 191
thrd_stack_alloc (C++ function), 195
thrd_stack_free (C++ function), 195
thrd_suspend (C++ function), 192
thrd_suspend_isr (C++ function), 195
thrd_t (C++ class), 197
thrd_t::elem (C++ member), 197
thrd_t::err (C++ member), 197
thrd_t::log_mask (C++ member), 197
thrd_t::name_p (C++ member), 197
thrd_t::next_p (C++ member), 197
thrd_t::port (C++ member), 197
thrd_t::prio (C++ member), 197
thrd_t::stack_size (C++ member), 197
thrd_t::state (C++ member), 197
thrd_t::thrd_p (C++ member), 197
thrd_yield (C++ function), 192
thrd_yield_isr (C++ function), 195
time (module), 197
time_add (C++ function), 198
time_busy_wait_us (C++ function), 198
time_get (C++ function), 197
time_set (C++ function), 197
time_subtract (C++ function), 198
time_t (C++ class), 198
time_t::nanoseconds (C++ member), 199
time_t::seconds (C++ member), 199
time_unix_time_to_date (C++ function), 198
timer (module), 199
timer_callback_t (C++ type), 200
timer_init (C++ function), 200
timer_module_init (C++ function), 200
TIMER_PERIODIC (C macro), 199
timer_start (C++ function), 200
timer_start_isr (C++ function), 200
timer_stop (C++ function), 200
timer_stop_isr (C++ function), 200
timer (C++ class), 201
timer::arg_p (C++ member), 201
timer::callback (C++ member), 201
timer::delta (C++ member), 201
timer::flags (C++ member), 201
timer::next_p (C++ member), 201
timer::timeout (C++ member), 201
TOKENPASTE (C macro), 201
TOKENPASTE2 (C macro), 201
TXC0 (C macro), 454
TXCIE0 (C macro), 454
TXEN0 (C macro), 454
types (module), 201

U
u16_t (C++ type), 202
U2X0 (C macro), 454
u32_t (C++ type), 202
u8_t (C++ type), 202
uart (module), 270
uart_0_dev (C macro), 448, 451
uart_1_dev (C macro), 448, 451
uart_2_dev (C macro), 448, 451
uart_device (C++ member), 272
UART_DEVICE_MAX (C macro), 452–457
uart_device_read (C++ function), 272
uart_device_start (C++ function), 272
uart_device_stop (C++ function), 272
uart_device_write (C++ function), 272
uart_init (C++ function), 271
uart_module_init (C++ function), 271
uart_read (C macro), 270
uart_rx_filter_cb_t (C++ type), 271
uart_set_rx_filterCb (C++ function), 271
uart_soft (module), 273
uart_soft_driver_t (C++ class), 273
uart_soft_driver_t::baudrate (C++ member), 274
uart_soft_driver_t::chout (C++ member), 274
uart_soft_driver_t::chout_exti (C++ member), 274
uart_soft_driver_t::rx_exti (C++ member), 274
uart_soft_driver_t::rx_pin (C++ member), 274
uart_soft_driver_t::sample_time (C++ member), 274
uart_soft_driver_t::tx_pin (C++ member), 274
uart_soft_init (C++ function), 273
uart_soft_read (C macro), 273
uart_soft_write (C macro), 273
uart_start (C++ function), 271
uart_stop (C++ function), 271
uart_write (C macro), 270
UCSZ00 (C macro), 454
UCSZ01 (C macro), 454
UCSZ02 (C macro), 454
UDRE0 (C macro), 454
UDRIE0 (C macro), 454
UNIQUE (C macro), 201
UNUSED (C macro), 201
UPE0 (C macro), 454
upgrade (module), 376
upgrade_application_enter (C++ function), 379
upgrade_application_erase (C++ function), 379
upgrade_application_is_valid (C++ function), 379
upgrade_binary_upload (C++ function), 379
upgrade_binary_upload_begin (C++ function), 379
upgrade_binary_upload_end (C++ function), 380
upgrade_bootloader_enter (C++ function), 379
upgrade_bootloader_clear (C++ function), 379
upgrade_bootloader_get (C++ function), 379
upgrade_bootloader_set (C++ function), 379
upgrade_module_init (C++ function), 379
UPM00 (C macro), 454
UPM01 (C macro), 454
USART0_RX_vect (C macro), 453
USART0_TX_vect (C macro), 453
USART0_UDRE_vect (C macro), 453
usb (module), 274
USB_CDC_CONTROL_LINE_STATE (C macro), 275
USB_CDC_LINE_CODING (C macro), 275
usb_cdc_line_info_t (C++ class), 281
usb_cdc_line_info_t::char_format (C++ member), 281
usb_cdc_line_info_t::data_bits (C++ member), 281
usb_cdc_line_info_t::dte_rate (C++ member), 281
usb_cdc_line_info_t::parity_type (C++ member), 281
USB_CDC_SEND_BREAK (C macro), 275
USB_CLASS_APPLICATION_SPECIFIC (C macro), 275
USB_CLASS_AUDIO (C macro), 275
USB_CLASS_AUDIO_VIDEO_DEVICES (C macro), 275
USB_CLASS_BILLBOARD_DEVICE_CLASS (C macro), 275
USB_CLASS_CDC_CONTROL (C macro), 275
USB_CLASS_CDC_DATA (C macro), 275
USB_CLASS_CONTENT_SECURITY (C macro), 275
USB_CLASS_DIAGNOSTIC_DEVICE (C macro), 275
USB_CLASS_HID (C macro), 275
USB_CLASS_HID_PROTOCOL_KEYBOARD (C macro), 286
USB_CLASS_HID_PROTOCOL_MOUSE (C macro), 286
USB_CLASS_HID_PROTOCOL_NONE (C macro), 286
USB_CLASS_HID_SUBCLASS_BOOT_INTERFACE (C macro), 286
USB_CLASS_HID_SUBCLASS_NONE (C macro), 286
USB_CLASS_HUB (C macro), 275
USB_CLASS_IMAGE (C macro), 275
USB_CLASS_MASS_STORAGE (C macro), 275
USB_CLASS_MISCELLANEOUS (C macro), 275
USB_CLASS_PERSONAL_HEALTHCARE (C macro), 275
USB_CLASS_PHYSICAL (C macro), 275
USB_CLASS_PRINTER (C macro), 275
USB_CLASS_SMART_CARD (C macro), 275
USB_CLASS_USE_INTERFACE (C macro), 275
USB_CLASS_VENDOR_SPECIFIC (C macro), 275
USB_CLASS_WIRELESS_CONTROLLER (C macro), 275
usb_desc_get_class (C++ function), 277
usb_desc_get_configuration (C++ function), 276
usb_desc_get_endpoint (C++ function), 276
usb_desc_get_interface (C++ function), 276
usb_descriptor_cdc_acm_t (C++ class), 280
usb_descriptor_cdc_acm_t::capabilities (C++ member), 280
usb_descriptor_cdc_acm_t::descriptor_type (C++ member), 280
usb_descriptor_cdc_acm_t::length (C++ member), 280
usb_descriptor_cdc_acm_t::sub_type (C++ member), 280
usb_descriptor_cdc_call_management_t (C++ class), 280
usb_descriptor_cdc_call_management_t::capabilities (C++ member), 280
usb_descriptor_cdc_call_management_t::data_interface (C++ member), 280
usb_descriptor_cdc_call_management_t::descriptor_type (C++ member), 280
usb_descriptor_cdc_call_management_t::length (C++ member), 280
usb_descriptor_cdc_call_management_t::sub_type (C++ member), 280
usb_descriptor_cdc_header_t (C++ class), 278
usb_descriptor_cdc_header_t::descriptor_type (C++ member), 278
usb_descriptor_cdc_union_t (C++ class), 278
usb_descriptor_cdc_union_t::descriptor_type (C++ member), 278
usb_descriptor_configuration_t (C++ class), 278
usb_descriptor_configuration_t::configuration (C++ member), 278
usb_descriptor_configuration_t::configuration_attributes (C++ member), 278
usb_descriptor_configuration_t::configuration_value (C++ member), 278
usb_descriptor_configuration_t::descriptor_type (C++ member), 278
usb_descriptor_configuration_t::length (C++ member), 278
usb_descriptor_configuration_t::max_power (C++ member), 279
usb_descriptor_configuration_t::num_interfaces (C++ member), 278
usb_descriptor_configuration_t::total_length (C++ member), 278
usb_descriptor_device_t (C++ class), 278
usb_descriptor_device_t::bcd_device (C++ member), 278
usb_descriptor_device_t::bcd_usb (C++ member), 278
usb_descriptor_device_t::descriptor_type (C++ member), 278
usb_descriptor_device_t::device_class (C++ member), 278
usb_descriptor_device_t::device_protocol (C++ member), 278
usb_descriptor_device_t::device_subclass (C++ member), 278
usb_descriptor_device_t::id_product (C++ member), 278
usb_descriptor_device_t::id_vendor (C++ member), 278
usb_descriptor_device_t::manufacturer (C++ member), 278
usb_descriptor_device_t::max_packet_size_0 (C++ member), 278
usb_descriptor_device_t::num_configurations (C++ member), 278
usb_descriptor_device_t::product (C++ member), 278
usb_descriptor_device_t::serial_number (C++ member), 278
usb_descriptor_endpoint_t (C++ class), 278
usb_descriptor_endpoint_t::attributes (C++ member), 278
usb_descriptor_endpoint_t::descriptor_type (C++ member), 278
usb_descriptor_endpoint_t::endpoint_address (C++ member), 278
usb_descriptor_endpoint_t::interval (C++ member), 279
usb_descriptor_endpoint_t::max_packet_size (C++ member), 279
usb_descriptor_endpoint_t::master_interface (C++ member), 279
usb_descriptor_endpoint_t::slave_interface (C++ member), 279
usb_descriptor_endpoint_t::sub_type (C++ member), 279
usb_descriptor_header_t (C++ class), 278
usb_descriptor_header_t::descriptor_type (C++ member), 278
usb_descriptor_header_t::length (C++ member), 278
usb_descriptor_interface_association_t (C++ class), 279
usb_descriptor_interface_association_t::descriptor_type (C++ member), 279
usb_descriptor_interface_association_t::first_interface (C++ member), 279
usb_descriptor_interface_association_t::function (C++ member), 280
usb_descriptor_interface_association_t::function_class (C++ member), 280
usb_descriptor_interface_association_t::function_protocol (C++ member), 280
usb_descriptor_interface_association_t::function_subclass (C++ member), 280
usb_descriptor_interface_association_t::interface_count (C++ member), 279
usb_descriptor_interface_t (C++ class), 279
usb_descriptor_interface_t::alternate_setting (C++ member), 279
usb_descriptor_interface_t::descriptor_type (C++ member), 279
usb_descriptor_interface_t::interface (C++ member), 279
usb_descriptor_interface_t::interface_class (C++ member), 279
usb_descriptor_interface_t::interface_number (C++ member), 279
usb_descriptor_interface_t::interface_protocol (C++ member), 279
usb_descriptor_interface_t::interfacesubclass (C++ member), 279
usb_descriptor_string_t (C++ class), 279
usb_descriptor_string_t::descriptor_type (C++ member), 279
usb_descriptor_string_t::length (C++ member), 279
usb_descriptor_string_t::string (C++ member), 279
usb_descriptor_t (C++ type), 283
usb_descriptor_t::configuration (C++ member), 281
usb_descriptor_t::device (C++ member), 281
usb_descriptor_t::endpoint (C++ member), 281
usb_descriptor_t::header (C++ member), 281
usb_descriptor_t::interface (C++ member), 281
usb_descriptor_t::string (C++ member), 281
usb_device (C++ member), 277
usb_device (module), 282
usb_device_class_cdc (module), 282
usb_device_class_cdc_driver_t (C++ class), 283
usb_device_class_cdc_driver_t::base (C++ member), 283
usb_device_class_cdc_driver_t::chin (C++ member), 283
usb_device_class_cdc_driver_t::chout (C++ member), 283
usb_device_class_cdc_driver_t::control_interface (C++ member), 283
usb_device_class_cdc_driver_t::drv_p (C++ member), 283
usb_device_class_cdc_driver_t::endpoint_in (C++ member), 283
usb_device_class_cdc_driver_t::endpoint_out (C++ member), 283
usb_device_class_cdc_driver_t::line_info (C++ member), 283
usb_device_class_cdc_driver_t::line_state (C++ member), 283
usb_device_class_cdc_init (C++ function), 283
usb_device_class_cdc_input_isr (C++ function), 283
usb_device_class_cdc_is_connected (C++ function), 283
usb_device_class_cdc_module_init (C++ function), 283
usb_device_class_cdc_read (C macro), 282
usb_device_class_cdc_write (C macro), 282
USB_DEVICE_MAX (C macro), 453
usb_device_module_init (C++ function), 284
usb_device_read_isr (C++ function), 285
usb_device_stop (C++ function), 284
usb_device_write (C++ function), 284
usb_device_write_isr (C++ function), 285
usb_format_descriptors (C++ function), 276
usb_host (module), 285
usb_host_class_hid (module), 285
usb_host_class_hid_device_t (C++ class), 286
usb_host_class_hid_device_t::buf (C++ member), 286
usb_host_class_hid_driver_t (C++ class), 286
usb_host_class_hid_driver_t::device_driver (C++ member), 286
usb_host_class_hid_driver_t::devices_p (C++ member), 287
usb_host_class_hid_device_t::length (C++ member), 287
usb_host_class_hid_device_t::size (C++ member), 287
usb_host_class_hid_init (C++ function), 286
usb_host_class_hid_list (C++ function), 286
usb_host_class_hid_stop (C++ function), 286
usb_host_class_mass_storage (module), 287
usb_host_class_mass_storage_device_read (C++ function), 287
usb_host_class_mass_storage_device_t (C++ class), 287
usb_host_class_mass_storage_device_t::buf (C++ member), 287
usb_host_class_mass_storage_driver_t (C++ class), 287
usb_host_class_mass_storage_driver_t::devices_p (C++ member), 288
usb_host_class_mass_storage_driver_t::devicetdriver (C++ member), 288
Index