WildhornAV

Generated by Doxygen 1.9.1

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 control Struct Reference

Data Fields

- control_state_t state

3.1.1 Field Documentation

3.1.1.1 state

```
control_state_t control::state
```

Referenced by control_abort_start(), control_apogee_start(), control_armed_start(), control_ballistic_start(), control_calibration_start(), control_coast_start(), control_drogue_start(), control_error_start(), control_event_↩ start(), control_idle_start(), control_main_start(), control_powered_start(), control_supersonic_start(), and control↩ _touchdown_start().

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

- control.c

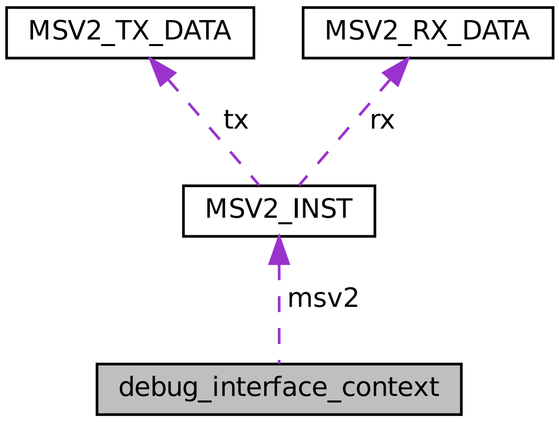
3.2 debug_context Struct Reference

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

- debug.c

3.3 debug_interface_context Struct Reference

Collaboration diagram for debug_interface_context:



Data Fields

- MSV2_INST_t msv2

3.3.1 Field Documentation

3.3.1.1 msv2

```
MSV2_INST_t debug_interface_context::msv2
```

Referenced by debug_init().

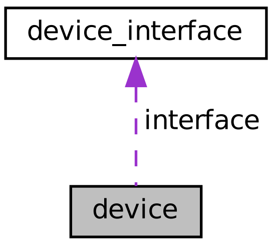
The documentation for this struct was generated from the following file:

- debug.c

3.4 device Struct Reference

```
#include <device.h>
```

Collaboration diagram for device:



Data Fields

- uint32_t id
- device_interface_t ∗ interface
- void ∗ context
- util_error_t(∗ read_reg)(void ∗, device_interface_t ∗, uint32_t, uint8_t ∗, uint32_t)
- util_error_t(∗ write_reg)(void ∗, device_interface_t ∗, uint32_t, uint8_t ∗, uint32_t)

3.4.1 Field Documentation

3.4.1.1 context

```
void* device::context
```

Referenced by device_create(), device_read_i16(), device_read_i32(), device_read_i8(), device_read_u16(), device_read_u32(), device_read_u8(), device_write_i16(), device_write_i32(), device_write_i8(), device_write_↩ u16(), device_write_u32(), and device_write_u8().

3.4.1.2 id

```
uint32_t device::id
```

Referenced by device_create().

3.4.1.3 interface

`device_interface_t* device::interface`

Referenced by device_create(), device_read_i16(), device_read_i32(), device_read_i8(), device_read_u16(), device_read_u32(), device_read_u8(), device_write_i16(), device_write_i32(), device_write_i8(), device_write_↩ u16(), device_write_u32(), and device_write_u8().

3.4.1.4 read_reg

`util_error_t(* device::read_reg) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t)`

Referenced by device_create(), device_read_i16(), device_read_i32(), device_read_i8(), device_read_u16(), device_read_u32(), and device_read_u8().

3.4.1.5 write_reg

`util_error_t(* device::write_reg) (void *, device_interface_t *, uint32_t, uint8_t *, uint32_t)`

Referenced by device_create(), device_write_i16(), device_write_i32(), device_write_i8(), device_write_u16(), device_write_u32(), and device_write_u8().

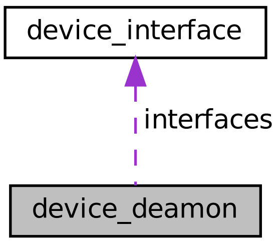
The documentation for this struct was generated from the following file:

- device.h

3.5 device_deamon Struct Reference

```
#include <device.h>
```

Collaboration diagram for device_deamon:



Data Fields

- uint32_t id
- StaticTask_t buffer
- StackType_t stack [DEAMON_STACK_SIZE]
- TaskHandle_t handle
- uint32_t interfaces_count
- device_interface_t ∗ interfaces [DEVICE_MAX_INTERFACES_PER_DEAMON]
- void ∗ context
- util_error_t(∗ data_rdy)(void ∗)

3.5.1 Field Documentation

3.5.1.1 buffer

```
StaticTask_t device_deamon::buffer
```

Referenced by device_deamon_create().

3.5.1.2 context

```
void* device_deamon::context
```

Referenced by device_deamon_create(), device_deamon_thread(), and HAL_UART_RxCpltCallback().

3.5.1.3 data_rdy

```
util_error_t(* device_deamon::data_rdy) (void *)
```

Referenced by device_deamon_create(), and device_deamon_thread().

3.5.1.4 handle

```
TaskHandle_t device_deamon::handle
```

Referenced by device_deamon_create().

3.5.1.5 id

```
uint32_t device_deamon::id
```

Referenced by device_deamon_create().

3.5.1.6 interfaces

```
device_interface_t* device_deamon::interfaces[DEVICE_MAX_INTERFACES_PER_DEAMON]
```

Referenced by device_interface_create(), and HAL_UART_RxCpltCallback().

3.5.1.7 interfaces_count

```
uint32_t device_deamon::interfaces_count
```

Referenced by device_deamon_create(), device_deamon_thread(), device_interface_create(), and HAL_UART_↩
RxCpltCallback().

3.5.1.8 stack

```
StackType_t device_deamon::stack[DEAMON_STACK_SIZE]
```

Referenced by device_deamon_create().

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

- device.h

3.6 device_interface Struct Reference

```
#include <device.h>
```

Data Fields

- uint32_t id
- void ∗ context
- util_error_t(∗ send)(void ∗, uint8_t ∗, uint32_t)
- util_error_t(∗ recv)(void ∗, uint8_t ∗, uint32_t ∗)
- util_error_t(∗ handle_data)(void ∗, void ∗)

3.6.1 Field Documentation

3.6.1.1 context

```
void* device_interface::context
```

Referenced by device_deamon_thread(), device_interface_create(), device_interface_recv(), device_interface_↩
send(), HAL_UART_RxCpltCallback(), read_reg(), and write_reg().

3.6.1.2 handle_data

```
util_error_t(* device_interface::handle_data) (void *, void *)
```

Referenced by device_deamon_thread(), and device_interface_create().

3.6.1.3 id

```
uint32_t device_interface::id
```

Referenced by device_interface_create().

3.6.1.4 recv

```
util_error_t(* device_interface::recv) (void *, uint8_t *, uint32_t *)
```

Referenced by device_interface_create(), and device_interface_recv().

3.6.1.5 send

```
util_error_t(* device_interface::send) (void *, uint8_t *, uint32_t)
```

Referenced by device_interface_create(), device_interface_recv(), and device_interface_send().

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

- device.h

3.7 dma_request Struct Reference

```
#include <dma.h>
```

Data Fields

- uint32_t src
- uint32_t dst
- uint32_t tranfser_len
- uint8_t dst_inc
- uint8_t src_inc

3.7.1 Field Documentation

3.7.1.1 dst

```
uint32_t dma_request::dst
```

3.7.1.2 dst_inc

```
uint8_t dma_request::dst_inc
```

3.7.1.3 src

```
uint32_t dma_request::src
```

3.7.1.4 src_inc

```
uint8_t dma_request::src_inc
```

3.7.1.5 tranfser_len

```
uint32_t dma_request::tranfser_len
```

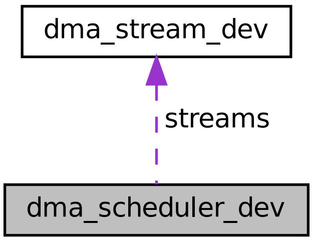
The documentation for this struct was generated from the following file:

- dma.h

3.8 dma_scheduler_dev Struct Reference

```
#include <dma.h>
```

Collaboration diagram for dma_scheduler_dev:



Data Fields

- uint16_t stream_count
- uint16_t free_stream_count
- dma_stream_dev_t ∗ streams [DMA_STREAMS_MAX_LEN]

3.8.1 Field Documentation

3.8.1.1 free_stream_count

```
uint16_t dma_scheduler_dev::free_stream_count
```

Referenced by dma_scheduler_release_stream(), and dma_scheduler_request_stream().

3.8.1.2 stream_count

```
uint16_t dma_scheduler_dev::stream_count
```

Referenced by dma_scheduler_request_stream().

3.8.1.3 streams

```
dma_stream_dev_t* dma_scheduler_dev::streams[DMA_STREAMS_MAX_LEN]
```

Referenced by dma_scheduler_init(), and dma_scheduler_request_stream().

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

- dma.h

3.9 dma_stream_config Struct Reference

```
#include <dma.h>
```

Data Fields

- uint32_t stream_number
- uint32_t p_addr
- uint32_t m0_addr
- uint32_t m1_addr
- uint32_t transfer_size
- uint16_t dmamux_request_number
- uint8_t priority
- dma_stream_dir_t direction
- uint8_t peripheral_flow_control
- void * user_context
- void(* transfer_cplt)(void *)
- void(* transfer_half)(void *)
- void(* transfer_error)(void *)

3.9.1 Field Documentation

3.9.1.1 direction

```
dma_stream_dir_t dma_stream_config::direction
```

Referenced by dma_start_stream().

3.9.1.2 dmamux_request_number

```
uint16_t dma_stream_config::dmamux_request_number
```

Referenced by dma_start_stream().

3.9.1.3 m0_addr

```
uint32_t dma_stream_config::m0_addr
```

Referenced by dma_start_stream().

3.9.1.4 m1_addr

```
uint32_t dma_stream_config::m1_addr
```

Referenced by dma_start_stream().

3.9.1.5 p_addr

```
uint32_t dma_stream_config::p_addr
```

Referenced by dma_start_stream().

3.9.1.6 peripheral_flow_control

```
uint8_t dma_stream_config::peripheral_flow_control
```

Referenced by dma_start_stream().

3.9.1.7 priority

```
uint8_t dma_stream_config::priority
```

Referenced by dma_start_stream().

3.9.1.8 stream_number

```
uint32_t dma_stream_config::stream_number
```

3.9.1.9 transfer_cplt

```
void(* dma_stream_config::transfer_cplt) (void *)
```

Referenced by dma_start_stream().

3.9.1.10 transfer_error

```
void(* dma_stream_config::transfer_error) (void *)
```

Referenced by dma_start_stream().

3.9.1.11 transfer_half

```
void(* dma_stream_config::transfer_half) (void *)
```

Referenced by dma_start_stream().

3.9.1.12 transfer_size

```
uint32_t dma_stream_config::transfer_size
```

Referenced by dma_start_stream().

3.9.1.13 user_context

```
void* dma_stream_config::user_context
```

Referenced by dma_start_stream().

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

- dma.h

3.10 dma_stream_dev Struct Reference

```
#include <dma.h>
```

Data Fields

- DMA_TypeDef ∗ dma
- DMA_Stream_TypeDef ∗ dma_stream
- DMAMUX_Channel_TypeDef ∗ dmamux_channel
- DMAMUX_ChannelStatus_TypeDef ∗ dmamux_channel_status
- dma_stream_state_t state
- uint16_t number
- void ∗ user_context
- void(∗ transfer_cplt)(void ∗)
- void(∗ transfer_half)(void ∗)
- void(∗ transfer_error)(void ∗)

3.10.1 Field Documentation

3.10.1.1 dma

```
DMA_TypeDef* dma_stream_dev::dma
```

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.1.2 dma_stream

```
DMA_Stream_TypeDef* dma_stream_dev::dma_stream
```

Referenced by dma_start_stream().

3.10.1.3 dmamux_channel

```
DMAMUX_Channel_TypeDef* dma_stream_dev::dmamux_channel
```

Referenced by dma_start_stream().

3.10.1.4 dmamux_channel_status

```
DMAMUX_ChannelStatus_TypeDef* dma_stream_dev::dmamux_channel_status
```

3.10.1.5 number

```
uint16_t dma_stream_dev::number
```

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.1.6 state

```
dma_stream_state_t dma_stream_dev::state
```

Referenced by dma_scheduler_init(), dma_scheduler_release_stream(), and dma_scheduler_request_stream().

3.10.1.7 transfer_cplt

```
void(* dma_stream_dev::transfer_cplt) (void *)
```

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.1.8 transfer_error

```
void(* dma_stream_dev::transfer_error) (void *)
```

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.1.9 transfer_half

```
void(* dma_stream_dev::transfer_half) (void *)
```

Referenced by dma_handle_interrupt(), and dma_start_stream().

3.10.1.10 user_context

```
void* dma_stream_dev::user_context
```

Referenced by dma_handle_interrupt(), and dma_start_stream().

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

- dma.h

3.11 gpio_config Struct Reference

```
#include <gpio.h>
```

Data Fields

- gpio_drive_t drive
- gpio_mode_t mode
- gpio_bias_t bias
- uint8_t speed
- uint8_t alternate

3.11.1 Field Documentation

3.11.1.1 alternate

```
uint8_t gpio_config::alternate
```

Referenced by gpio_cfg().

3.11.1.2 bias

```
gpio_bias_t gpio_config::bias
```

3.11.1.3 drive

```
gpio_drive_t gpio_config::drive
```

Referenced by gpio_cfg().

3.11.1.4 mode

```
gpio_mode_t gpio_config::mode
```

Referenced by gpio_cfg().

3.11.1.5 speed

```
uint8_t gpio_config::speed
```

Referenced by gpio_cfg().

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

- gpio.h

3.12 hostproc_interface_context Struct Reference

Data Fields

- VIRT_UART_HandleTypeDef ∗ uart
- uint8_t rx_once

3.12.1 Field Documentation

3.12.1.1 rx_once

```
uint8_t hostproc_interface_context::rx_once
```

Referenced by host_send(), host_UART0_RX(), and hostproc_init().

3.12.1.2 uart

```
VIRT_UART_HandleTypeDef* hostproc_interface_context::uart
```

Referenced by host_send(), and hostproc_init().

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

- hostproc.c

3.13 i2c_interface_context Struct Reference

I2C interface context structure.

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

Data Fields

- I2C_HandleTypeDef ∗ i2c

3.13.1 Detailed Description

I2C interface context structure.

this only contains the HAL I2C handle pointer.

3.13.2 Field Documentation

3.13.2.1 i2c

```
I2C_HandleTypeDef* i2c_interface_context::i2c
```

Referenced by read_reg(), and write_reg().

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

- i2c.h

3.14 i2c_sensor_context Struct Reference

Data Fields

- uint8_t device_address

3.14.1 Field Documentation

3.14.1.1 device_address

```
uint8_t i2c_sensor_context::device_address
```

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

- i2c_sensor.c

3.15 led_color Struct Reference

```
#include <led.h>
```

Data Fields

- uint8_t r
- uint8_t g
- uint8_t b

3.15.1 Field Documentation

3.15.1.1 b

```
uint8_t led_color::b
```

Referenced by led_rgb_set_color().

3.15.1.2 g

```
uint8_t led_color::g
```

Referenced by led_rgb_set_color().

3.15.1.3 r

```
uint8_t led_color::r
```

Referenced by led_rgb_set_color().

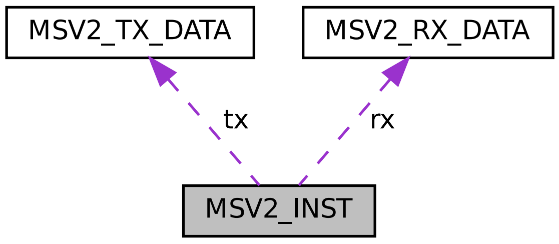
The documentation for this struct was generated from the following file:

- led.h

3.16 MSV2_INST Struct Reference

```
#include <msv2.h>
```

Collaboration diagram for MSV2_INST:



Data Fields

- uint32_t id
- MSV2_RX_DATA_t rx
- MSV2_TX_DATA_t tx

3.16.1 Field Documentation

3.16.1.1 id

```
uint32_t MSV2_INST::id
```

Referenced by msv2_init().

3.16.1.2 rx

```
MSV2_RX_DATA_t MSV2_INST::rx
```

Referenced by msv2_decode_fragment(), and msv2_rx_data().

3.16.1.3 tx

```
MSV2_TX_DATA_t MSV2_INST::tx
```

Referenced by msv2_create_frame(), and msv2_tx_data().

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

- msv2.h

3.17 MSV2_RX_DATA Struct Reference

```
#include <msv2.h>
```

Data Fields

- uint8_t opcode
- uint8_t data_len
- uint16_t crc
- MSV2_DECODE_STATE_t state
- uint8_t escape
- uint16_t length
- uint16_t counter
- uint8_t data [MSV2_MAX_FRAME_LEN]
- uint16_t crc_data [MSV2_MAX_FRAME_LEN/sizeof(uint16_t)]

3.17.1 Field Documentation

3.17.1.1 counter

```
uint16_t MSV2_RX_DATA::counter
```

Referenced by msv2_decode_fragment().

3.17.1.2 crc

```
uint16_t MSV2_RX_DATA::crc
```

Referenced by msv2_decode_fragment().

3.17.1.3 crc_data

```
uint16_t MSV2_RX_DATA::crc_data[MSV2_MAX_FRAME_LEN/sizeof(uint16_t)]
```

Referenced by msv2_decode_fragment().

3.17.1.4 data

```
uint8_t MSV2_RX_DATA::data[MSV2_MAX_FRAME_LEN]
```

Referenced by msv2_decode_fragment(), and msv2_rx_data().

3.17.1.5 data_len

```
uint8_t MSV2_RX_DATA::data_len
```

Referenced by msv2_decode_fragment().

3.17.1.6 escape

```
uint8_t MSV2_RX_DATA::escape
```

Referenced by msv2_decode_fragment().

3.17.1.7 length

```
uint16_t MSV2_RX_DATA::length
```

Referenced by msv2_decode_fragment().

3.17.1.8 opcode

```
uint8_t MSV2_RX_DATA::opcode
```

Referenced by msv2_decode_fragment().

3.17.1.9 state

```
MSV2_DECODE_STATE_t MSV2_RX_DATA::state
```

Referenced by msv2_decode_fragment().

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

- msv2.h

3.18 MSV2_TX_DATA Struct Reference

```
#include <msv2.h>
```

Data Fields

- uint8_t opcode
- uint8_t data_len
- uint16_t crc
- uint8_t data [MSV2_MAX_FRAME_LEN]
- uint16_t crc_data [MSV2_MAX_FRAME_LEN/sizeof(uint16_t)]

3.18.1 Field Documentation

3.18.1.1 crc

```
uint16_t MSV2_TX_DATA::crc
```

3.18.1.2 crc_data

```
uint16_t MSV2_TX_DATA::crc_data[MSV2_MAX_FRAME_LEN/sizeof(uint16_t)]
```

Referenced by msv2_create_frame().

3.18.1.3 data

```
uint8_t MSV2_TX_DATA::data[MSV2_MAX_FRAME_LEN]
```

Referenced by msv2_create_frame(), and msv2_tx_data().

3.18.1.4 data_len

```
uint8_t MSV2_TX_DATA::data_len
```

Referenced by msv2_create_frame().

3.18.1.5 opcode

```
uint8_t MSV2_TX_DATA::opcode
```

Referenced by msv2_create_frame().

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

- msv2.h

3.19 note Struct Reference

```
#include <note.h>
```

Data Fields

- uint16_t freq
- uint16_t time

3.19.1 Field Documentation

3.19.1.1 freq

```
uint16_t note::freq
```

3.19.1.2 time

```
uint16_t note::time
```

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

- note.h

3.20 od_entry_t Struct Reference

Data Fields

- uint8_t data_id
- uint8_t size
- uint8_t ∗ data

3.20.1 Field Documentation

3.20.1.1 data

```
uint8_t* od_entry_t::data
```

Referenced by od_unsafe_read(), and od_update_task().

3.20.1.2 data_id

```
uint8_t od_entry_t::data_id
```

Referenced by od_unsafe_write().

3.20.1.3 size

```
uint8_t od_entry_t::size
```

Referenced by od_unsafe_read(), od_unsafe_write(), and od_update_task().

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

- od.c

3.21 od_frame_t Struct Reference

Data Fields

- uint8_t data_id
- uint8_t size
- uint8_t data [OD_FRAME_MAX_SIZE]

3.21.1 Field Documentation

3.21.1.1 data

```
uint8_t od_frame_t::data[OD_FRAME_MAX_SIZE]
```

Referenced by od_unsafe_write(), and od_update_task().

3.21.1.2 data_id

```
uint8_t od_frame_t::data_id
```

Referenced by od_unsafe_write(), and od_update_task().

3.21.1.3 size

```
uint8_t od_frame_t::size
```

Referenced by od_unsafe_write().

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

- od.c

3.22 packet_def Struct Reference

```
#include <packet.h>
```

Data Fields

- uint8_t opcode
- uint8_t len

3.22.1 Field Documentation

3.22.1.1 len

```
uint8_t packet_def::len
```

3.22.1.2 opcode

```
uint8_t packet_def::opcode
```

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

- packet.h

3.23 serial_deamon_context Struct Reference

```
#include <serial.h>
```

Data Fields

- SemaphoreHandle_t rx_sem
- StaticSemaphore_t rx_sem_buffer

3.23.1 Field Documentation

3.23.1.1 rx_sem

```
SemaphoreHandle_t serial_deamon_context::rx_sem
```

Referenced by HAL_UART_RxCpltCallback(), serial_data_ready(), and serial_init().

3.23.1.2 rx_sem_buffer

```
StaticSemaphore_t serial_deamon_context::rx_sem_buffer
```

Referenced by serial_init().

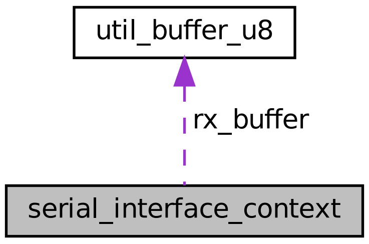
The documentation for this struct was generated from the following file:

- serial.h

3.24 serial_interface_context Struct Reference

```
#include <serial.h>
```

Collaboration diagram for serial_interface_context:



Data Fields

- UART_HandleTypeDef ∗ uart
- util_buffer_u8_t rx_buffer
- uint8_t rx_data [SERIAL_BUFFER_LEN]
- uint32_t rx_data_len
- uint8_t rx_fragment
- uint8_t tx_data [SERIAL_BUFFER_LEN]
- void ∗ protocol

3.24.1 Field Documentation

3.24.1.1 protocol

```
void* serial_interface_context::protocol
```

3.24.1.2 rx_buffer

```
util_buffer_u8_t serial_interface_context::rx_buffer
```

Referenced by HAL_UART_RxCpltCallback(), serial_recv(), and serial_setup_reception().

3.24.1.3 rx_data

```
uint8_t serial_interface_context::rx_data[SERIAL_BUFFER_LEN]
```

Referenced by serial_setup_reception().

3.24.1.4 rx_data_len

```
uint32_t serial_interface_context::rx_data_len
```

3.24.1.5 rx_fragment

```
uint8_t serial_interface_context::rx_fragment
```

Referenced by HAL_UART_RxCpltCallback(), and serial_setup_reception().

3.24.1.6 tx_data

```
uint8_t serial_interface_context::tx_data[SERIAL_BUFFER_LEN]
```

3.24.1.7 uart

```
UART_HandleTypeDef* serial_interface_context::uart
```

Referenced by HAL_UART_RxCpltCallback(), serial_send(), and serial_setup_reception().

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

- serial.h

3.25 util_buffer_i16 Struct Reference

```
#include <util.h>
```

Data Fields

- uint16_t c_ix
- uint16_t l_ix
- uint16_t bfr_len
- int16_t * buffer

3.25.1 Field Documentation

3.25.1.1 bfr_len

```
uint16_t util_buffer_i16::bfr_len
```

Referenced by util_buffer_i16_add(), util_buffer_i16_get(), and util_buffer_i16_init().

3.25.1.2 buffer

```
int16_t* util_buffer_i16::buffer
```

Referenced by util_buffer_i16_add(), util_buffer_i16_get(), and util_buffer_i16_init().

3.25.1.3 c_ix

```
uint16_t util_buffer_i16::c_ix
```

Referenced by util_buffer_i16_add(), util_buffer_i16_init(), and util_buffer_i16_isempty().

3.25.1.4 l_ix

```
uint16_t util_buffer_i16::l_ix
```

Referenced by util_buffer_i16_get(), util_buffer_i16_init(), and util_buffer_i16_isempty().

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

- util.h

3.26 util_buffer_u16 Struct Reference

```
#include <util.h>
```

Data Fields

- uint16_t c_ix
- uint16_t l_ix
- uint16_t bfr_len
- uint16_t * buffer

3.26.1 Field Documentation

3.26.1.1 bfr_len

```
uint16_t util_buffer_u16::bfr_len
```

Referenced by util_buffer_u16_add(), util_buffer_u16_get(), and util_buffer_u16_init().

3.26.1.2 buffer

```
uint16_t* util_buffer_u16::buffer
```

Referenced by util_buffer_u16_add(), util_buffer_u16_get(), and util_buffer_u16_init().

3.26.1.3 c_ix

```
uint16_t util_buffer_u16::c_ix
```

Referenced by util_buffer_u16_add(), util_buffer_u16_init(), and util_buffer_u16_isempty().

3.26.1.4 l_ix

```
uint16_t util_buffer_u16::l_ix
```

Referenced by util_buffer_u16_get(), util_buffer_u16_init(), and util_buffer_u16_isempty().

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

- util.h

3.27 util_buffer_u8 Struct Reference

```
#include <util.h>
```

Data Fields

- uint16_t c_ix
- uint16_t l_ix
- uint16_t bfr_len
- uint8_t ∗ buffer

3.27.1 Field Documentation

3.27.1.1 bfr_len

```
uint16_t util_buffer_u8::bfr_len
```

Referenced by util_buffer_u8_access(), util_buffer_u8_add(), util_buffer_u8_get(), and util_buffer_u8_init().

3.27.1.2 buffer

```
uint8_t* util_buffer_u8::buffer
```

Referenced by util_buffer_u8_access(), util_buffer_u8_add(), util_buffer_u8_get(), and util_buffer_u8_init().

3.27.1.3 c_ix

```
uint16_t util_buffer_u8::c_ix
```

Referenced by util_buffer_u8_access(), util_buffer_u8_add(), util_buffer_u8_init(), and util_buffer_u8_isempty().

3.27.1.4 l_ix

```
uint16_t util_buffer_u8::l_ix
```

Referenced by util_buffer_u8_get(), util_buffer_u8_init(), and util_buffer_u8_isempty().

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

- util.h

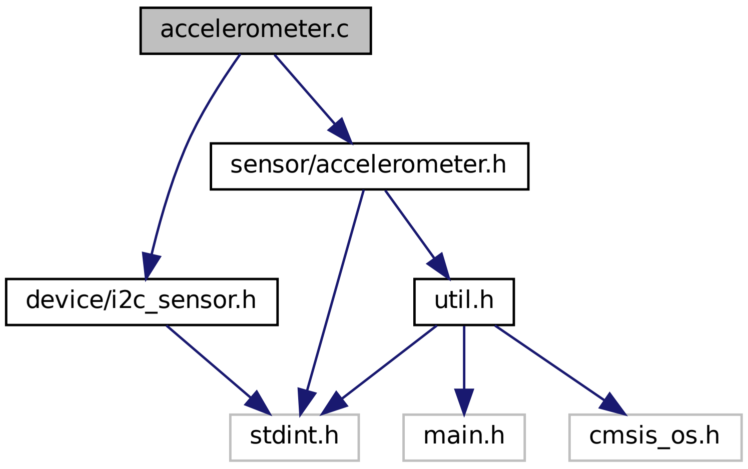
Chapter 4

File Documentation

4.1 accelerometer.c File Reference

```
#include <sensor/accelerometer.h>
#include <device/i2c_sensor.h>
```

Include dependency graph for accelerometer.c:



Functions

- util_error_t accelerometer_init (void)

  Initialize accelerometers.

4.1.1 Function Documentation

4.1.1.1 accelerometer_init()

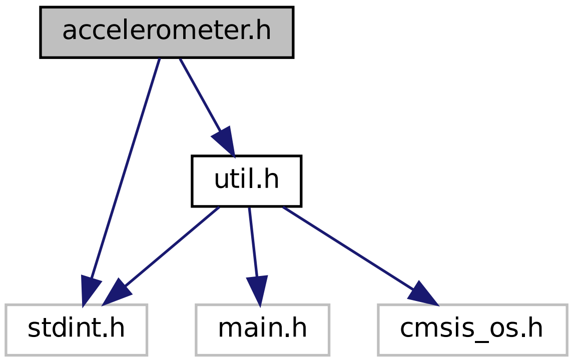
```
util_error_t accelerometer_init (
             void  )
```

Initialize accelerometers.

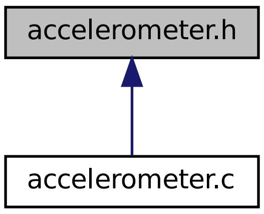
4.2 accelerometer.h File Reference

```
#include <stdint.h>
#include <util.h>
```

Include dependency graph for accelerometer.h:



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



Functions

- util_error_t accelerometer_init (void)

  Initialize accelerometers.

4.2.1 Function Documentation

4.2.1.1 accelerometer_init()

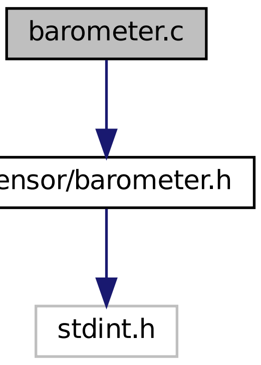
```
util_error_t accelerometer_init (
            void  )
```

Initialize accelerometers.

4.3 barometer.c File Reference

```
#include <sensor/barometer.h>
```

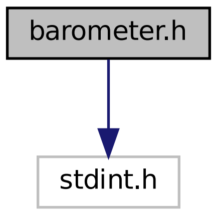
Include dependency graph for barometer.c:



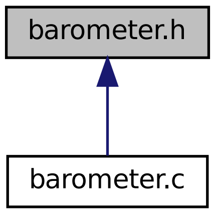
4.4 barometer.h File Reference

```
#include <stdint.h>
```

Include dependency graph for barometer.h:



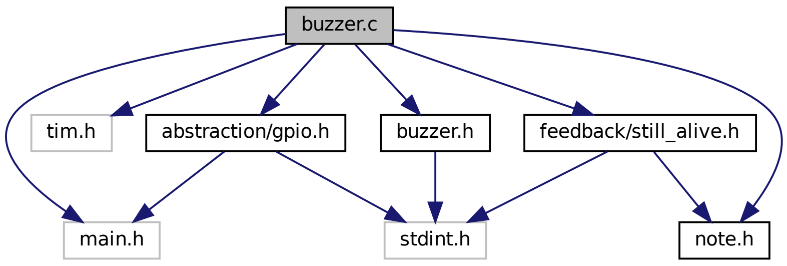
This graph shows which files directly or indirectly include this file:



4.5 buzzer.c File Reference

```
#include <main.h>
#include <tim.h>
#include "buzzer.h"
#include <abstraction/gpio.h>
#include <feedback/still_alive.h>
#include "note.h"
```

Include dependency graph for buzzer.c:



Macros

- #define NOTE_TIMER_DEV htim14
- #define RYTM_TIMER_DEV htim16
- #define NOTE_TIMER NOTE_TIMER_DEV.Instance
- #define RYTM_TIMER RYTM_TIMER_DEV.Instance
- #define BUZZER_PIN GPIO_PIN_3
- #define BUZZER_PORT GPIOC
- #define TIMER_FREQ 200e6
- #define NOTE_PRESC 10
- #define RYTM_PRESC 20000
- #define TIMER_TRIM -1e6
- #define COMPUTE_NOTE(note) ((((TIMER_FREQ)+(TIMER_TRIM))∗10)/(NOTE_PRESC)/(note))/2
- #define RYTM_MS(ms) (ms)∗(((TIMER_FREQ)+(TIMER_TRIM))/(RYTM_PRESC))/(1000)
- #define COMPUTE_RYTM(time) RYTM_MS((time)∗100)

Functions

- void buzzer_note_interrupt (void)
- void buzzer_rytm_interrupt (void)
- void buzzer_enable (void)
- void buzzer_disable (void)
- void buzzer_init (void)

Variables

- static uint16_t melody_state = 0
- static uint8_t state = 0
- static uint8_t melody_active = 1

4.5.1 Macro Definition Documentation

4.5.1.1 BUZZER_PIN

```
#define BUZZER_PIN GPIO_PIN_3
```

4.5.1.2 BUZZER_PORT

```
#define BUZZER_PORT GPIOC
```

4.5.1.3 COMPUTE_NOTE

```
#define COMPUTE_NOTE(
              note ) ((((TIMER_FREQ)+(TIMER_TRIM))*10)/(NOTE_PRESC)/(note))/2
```

4.5.1.4 COMPUTE_RYTM

```
#define COMPUTE_RYTM(
              time ) RYTM_MS((time)*100)
```

4.5.1.5 NOTE_PRESC

```
#define NOTE_PRESC 10
```

4.5.1.6 NOTE_TIMER

```
#define NOTE_TIMER NOTE_TIMER_DEV.Instance
```

4.5.1.7 NOTE_TIMER_DEV

```
#define NOTE_TIMER_DEV htim14
```

4.5.1.8 RYTM_MS

```
#define RYTM_MS(
              ms ) (ms)*(((TIMER_FREQ)+(TIMER_TRIM))/(RYTM_PRESC))/(1000)
```

4.5.1.9 RYTM_PRESC

```
#define RYTM_PRESC 20000
```

4.5.1.10 RYTM_TIMER

```
#define RYTM_TIMER RYTM_TIMER_DEV.Instance
```

4.5.1.11 RYTM_TIMER_DEV

```
#define RYTM_TIMER_DEV htim16
```

4.5.1.12 TIMER_FREQ

```
#define TIMER_FREQ 200e6
```

4.5.1.13 TIMER_TRIM

```
#define TIMER_TRIM -1e6
```

4.5.2 Function Documentation

4.5.2.1 buzzer_disable()

```
void buzzer_disable (
            void )
```

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

4.5.2.2 buzzer_enable()

```
void buzzer_enable (
            void )
```

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

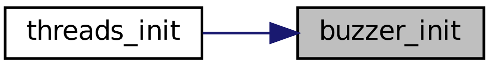
4.5.2.3 buzzer_init()

```
void buzzer_init (
            void )
```

References A4, BUZZER_PIN, BUZZER_PORT, COMPUTE_NOTE, COMPUTE_RYTM, melody_active, NOTE_↩
TIMER, and RYTM_TIMER.

Referenced by threads_init().

Here is the caller graph for this function:

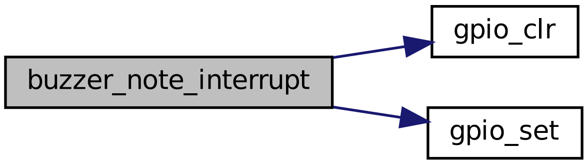


4.5.2.4 buzzer_note_interrupt()

```
void buzzer_note_interrupt (
             void  )
```

References BUZZER_PIN, BUZZER_PORT, gpio_clr(), gpio_set(), melody_active, and state.

Here is the call graph for this function:

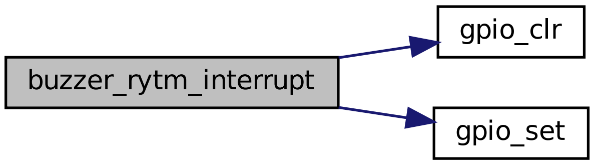


4.5.2.5 buzzer_rytm_interrupt()

```
void buzzer_rytm_interrupt (
             void  )
```

References COMPUTE_NOTE, gpio_clr(), gpio_set(), melody_active, melody_state, NOTE_TIMER, still_alive, and still_alive_len.

Here is the call graph for this function:



4.5.3 Variable Documentation

4.5.3.1 melody_active

```
uint8_t melody_active = 1  [static]
```

Referenced by buzzer_init(), buzzer_note_interrupt(), and buzzer_rytm_interrupt().

4.5.3.2 melody_state

```
uint16_t melody_state = 0  [static]
```

Referenced by buzzer_rytm_interrupt().

4.5.3.3 state

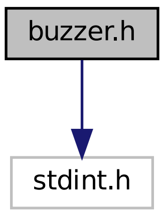
```
uint8_t state = 0  [static]
```

Referenced by buzzer_note_interrupt().

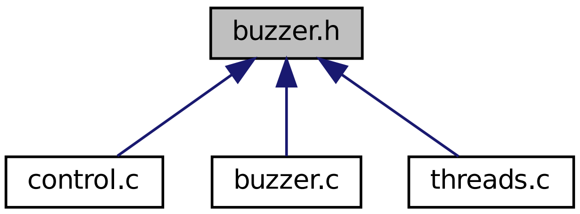
4.6 buzzer.h File Reference

```
#include <stdint.h>
```

Include dependency graph for buzzer.h:



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



Functions

- void buzzer_note_interrupt (void)
- void buzzer_rytm_interrupt (void)
- void buzzer_enable (void)
- void buzzer_disable (void)
- void buzzer_init (void)

4.6.1 Function Documentation

4.6.1.1 buzzer_disable()

```
void buzzer_disable (
             void  )
```

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

4.6.1.2 buzzer_enable()

```
void buzzer_enable (
             void  )
```

References NOTE_TIMER_DEV, and RYTM_TIMER_DEV.

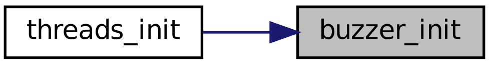
4.6.1.3 buzzer_init()

```
void buzzer_init (
             void  )
```

References A4, BUZZER_PIN, BUZZER_PORT, COMPUTE_NOTE, COMPUTE_RYTM, melody_active, NOTE_↩ TIMER, and RYTM_TIMER.

Referenced by threads_init().

Here is the caller graph for this function:

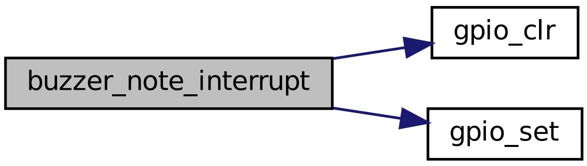


4.6.1.4 buzzer_note_interrupt()

```
void buzzer_note_interrupt (
             void   )
```

References BUZZER_PIN, BUZZER_PORT, gpio_clr(), gpio_set(), melody_active, and state.

Here is the call graph for this function:

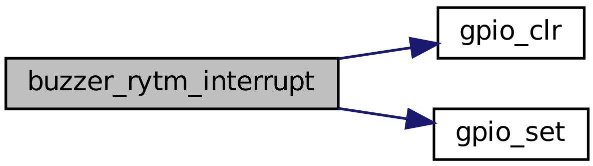


4.6.1.5 buzzer_rytm_interrupt()

```
void buzzer_rytm_interrupt (
             void   )
```

References COMPUTE_NOTE, gpio_clr(), gpio_set(), melody_active, melody_state, NOTE_TIMER, still_alive, and still_alive_len.

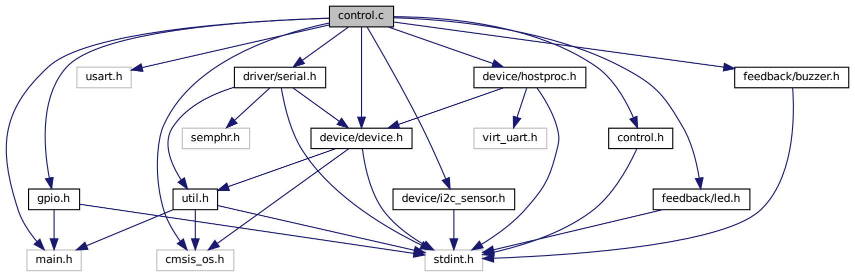
Here is the call graph for this function:



4.7 control.c File Reference

```
#include <main.h>
#include <gpio.h>
#include <usart.h>
#include <cmsis_os.h>
#include <driver/serial.h>
#include <device/device.h>
#include <device/i2c_sensor.h>
#include <device/hostproc.h>
#include <control.h>
#include <feedback/led.h>
#include <feedback/buzzer.h>
```

Include dependency graph for control.c:



Data Structures

- struct control

Macros

- #define CONTROL_HEART_BEAT 200

Typedefs

- typedef enum control_state control_state_t

  State of the control FSM.
- typedef struct control control_t

Enumerations

- enum control_state {
  CONTROL_IDLE , CONTROL_CALIBRATION , CONTROL_ARMED , CONTROL_POWERED ,
  CONTROL_SUPERSONIC , CONTROL_COAST , CONTROL_APOGEE , CONTROL_DROGUE ,
  CONTROL_EVENT , CONTROL_MAIN , CONTROL_TOUCHDOWN , CONTROL_BALLISTIC ,
  CONTROL_ERROR , CONTROL_ABORT }

  State of the control FSM.

Functions

- void control_idle_start (void)

  Idle state entry.
- void control_idle_run (void)

  Idle state runtime.
- void control_calibration_start (void)

  Calibration state entry.
- void control_calibration_run (void)

  Calibration state runtime.
- void control_armed_start (void)

  Armed state entry.
- void control_armed_run (void)

  Armed state runtime.
- void control_powered_start (void)

  Powered state entry.
- void control_powered_run (void)

  Powered state runtime.
- void control_supersonic_start (void)

  Supersonic state entry.
- void control_supersonic_run (void)

  Supersonic state runtime.
- void control_coast_start (void)

  Coast state entry.
- void control_coast_run (void)

  Coast state runtime.
- void control_apogee_start (void)

  Apogee state entry.
- void control_apogee_run (void)

  Apogee state runtime.
- void control_drogue_start (void)

  Drogue state entry.
- void control_drogue_run (void)

  Drogue state runtime.
- void control_event_start (void)

  Event state entry.
- void control_event_run (void)

  Event state runtime.
- void control_main_start (void)

  Main state entry.
- void control_main_run (void)

  Main state runtime.
- void control_touchdown_start (void)

  Touchdown state entry.
- void control_touchdown_run (void)

  Touchdown state runtime.
- void control_ballistic_start (void)

  Ballistic state entry.
- void control_ballistic_run (void)

  Ballistic state runtime.
- void control_error_start (void)

Error state entry.
- void control_error_run (void)

  Error state runtime.
- void control_abort_start (void)

  Abort state entry.
- void control_abort_run (void)

  Abort state runtime.
- void control_thread (__attribute__((unused)) void ∗arg)

  Control thread entry point.

Variables

- control_t control

4.7.1 Macro Definition Documentation

4.7.1.1 CONTROL_HEART_BEAT

```
#define CONTROL_HEART_BEAT 200
```

4.7.2 Typedef Documentation

4.7.2.1 control_state_t

```
typedef enum control_state control_state_t
```

State of the control FSM.

4.7.2.2 control_t

```
typedef struct control control_t
```

4.7.3 Enumeration Type Documentation

4.7.3.1 control_state

```
enum control_state
```

State of the control FSM.

Enumerator

CONTROL_IDLE	Wait for arming or calibration
CONTROL_CALIBRATION	Calibrate sensors and actuators
CONTROL_ARMED	Armed, wait for liftoff
CONTROL_POWERED	Powered ascent
CONTROL_SUPERSONIC	Supersonic flight
CONTROL_COAST	Subsonic, coast flight
CONTROL_APOGEE	Apogee reached, trigger first event
CONTROL_DROGUE	Drogue chute descent, wait for second event
CONTROL_EVENT	Low alt reached, trigger second event
CONTROL_MAIN	Main chute descent, wait for touchdown
CONTROL_TOUCHDOWN	Touchdown detected, end of the flight
CONTROL_BALLISTIC	Ballistic flight detected
CONTROL_ERROR	Auto triggered error
CONTROL_ABORT	User triggered error

4.7.4 Function Documentation

4.7.4.1 control_abort_run()

```
void control_abort_run (
             void  )
```

Abort state runtime.

4.7.4.2 control_abort_start()

```
void control_abort_start (
             void  )
```

Abort state entry.

References CONTROL_ABORT, and control::state.

4.7.4.3 control_apogee_run()

```
void control_apogee_run (
             void  )
```

Apogee state runtime.

4.7.4.4 control_apogee_start()

```
void control_apogee_start (
             void  )
```

Apogee state entry.

References CONTROL_APOGEE, and control::state.

4.7.4.5 control_armed_run()

```
void control_armed_run (
             void  )
```

Armed state runtime.

4.7.4.6 control_armed_start()

```
void control_armed_start (
             void  )
```

Armed state entry.

References CONTROL_ARMED, and control::state.

4.7.4.7 control_ballistic_run()

```
void control_ballistic_run (
             void  )
```

Ballistic state runtime.

4.7.4.8 control_ballistic_start()

```
void control_ballistic_start (
             void  )
```

Ballistic state entry.

References CONTROL_BALLISTIC, and control::state.

4.7.4.9 control_calibration_run()

```
void control_calibration_run (
            void  )
```

Calibration state runtime.

4.7.4.10 control_calibration_start()

```
void control_calibration_start (
            void  )
```

Calibration state entry.

References CONTROL_CALIBRATION, and control::state.

4.7.4.11 control_coast_run()

```
void control_coast_run (
            void  )
```

Coast state runtime.

4.7.4.12 control_coast_start()

```
void control_coast_start (
            void  )
```

Coast state entry.

References CONTROL_COAST, and control::state.

4.7.4.13 control_drogue_run()

```
void control_drogue_run (
            void  )
```

Drogue state runtime.

4.7.4.14 control_drogue_start()

```
void control_drogue_start (
            void  )
```

Drogue state entry.

References CONTROL_DROGUE, and control::state.

4.7.4.15 control_error_run()

```
void control_error_run (
            void  )
```

Error state runtime.

4.7.4.16 control_error_start()

```
void control_error_start (
            void  )
```

Error state entry.

References CONTROL_ERROR, and control::state.

4.7.4.17 control_event_run()

```
void control_event_run (
            void  )
```

Event state runtime.

4.7.4.18 control_event_start()

```
void control_event_start (
            void  )
```

Event state entry.

References CONTROL_EVENT, and control::state.

4.7.4.19 control_idle_run()

```
void control_idle_run (
            void )
```

Idle state runtime.

4.7.4.20 control_idle_start()

```
void control_idle_start (
            void )
```

Idle state entry.

References CONTROL_IDLE, and control::state.

4.7.4.21 control_main_run()

```
void control_main_run (
            void )
```

Main state runtime.

4.7.4.22 control_main_start()

```
void control_main_start (
            void )
```

Main state entry.

References CONTROL_MAIN, and control::state.

4.7.4.23 control_powered_run()

```
void control_powered_run (
            void )
```

Powered state runtime.

4.7.4.24 control_powered_start()

```
void control_powered_start (
            void  )
```

Powered state entry.

References CONTROL_POWERED, and control::state.

4.7.4.25 control_supersonic_run()

```
void control_supersonic_run (
            void  )
```

Supersonic state runtime.

4.7.4.26 control_supersonic_start()

```
void control_supersonic_start (
            void  )
```

Supersonic state entry.

References CONTROL_SUPERSONIC, and control::state.

4.7.4.27 control_thread()

```
void control_thread (
            __attribute__((unused)) void * arg )
```

Control thread entry point.

This thread holds the main state machine of the WildhornAV software. It will be the main decision point for actions to be taken with respect to real world events.

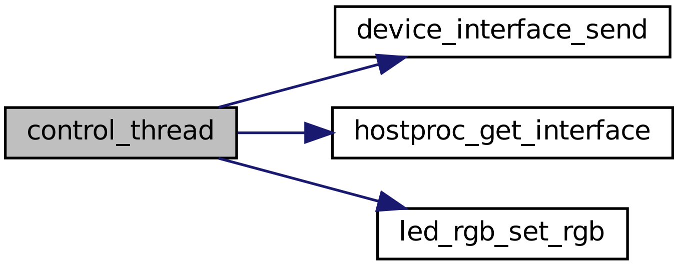
Parameters

arg	freertos thread entry point context (unused)

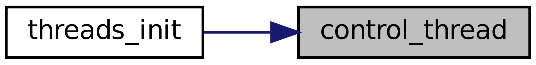
References CONTROL_HEART_BEAT, device_interface_send(), hostproc_get_interface(), hostproc_interface, and led_rgb_set_rgb().

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.4.28 control_touchdown_run()

```
void control_touchdown_run (
             void  )
```

Touchdown state runtime.

4.7.4.29 control_touchdown_start()

```
void control_touchdown_start (
             void  )
```

Touchdown state entry.

References CONTROL_TOUCHDOWN, and control::state.

4.7.5 Variable Documentation

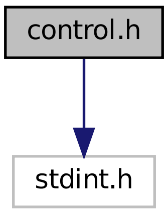
4.7.5.1 control

```
control_t control
```

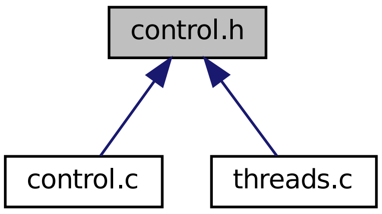
4.8 control.h File Reference

```
#include <stdint.h>
```

Include dependency graph for control.h:



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



Functions

- void control_thread (void ∗arg)

4.8.1 Function Documentation

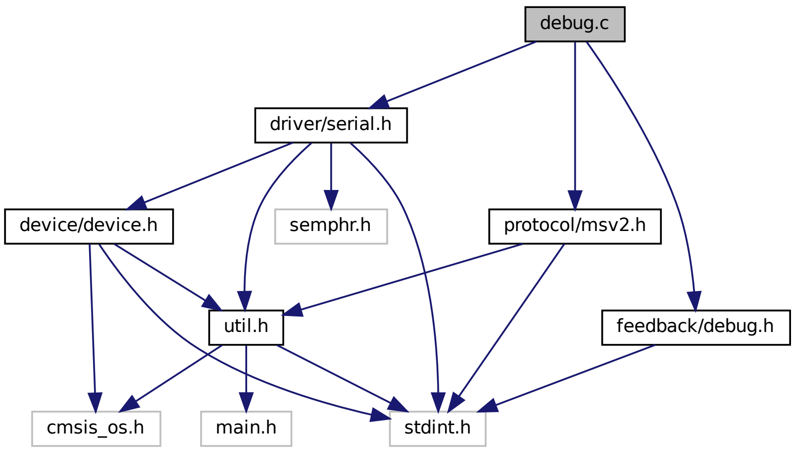
4.8.1.1 control_thread()

```
void control_thread (
             void * arg )
```

4.9 debug.c File Reference

```
#include <protocol/msv2.h>
#include <feedback/debug.h>
#include <driver/serial.h>
```

Include dependency graph for debug.c:



Data Structures

- struct debug_interface_context
- struct debug_context

Typedefs

- typedef struct debug_interface_context debug_interface_context_t
- typedef struct debug_context debug_context_t

Functions

- util_error_t debug_init (void)

Variables

- debug_context_t debug_context
- serial_interface_context_t feedback_interface_context
- debug_interface_context_t debug_interface_context

4.9.1 Typedef Documentation

4.9.1.1 debug_context_t

```
typedef struct debug_context debug_context_t
```

4.9.1.2 debug_interface_context_t

```
typedef struct debug_interface_context debug_interface_context_t
```

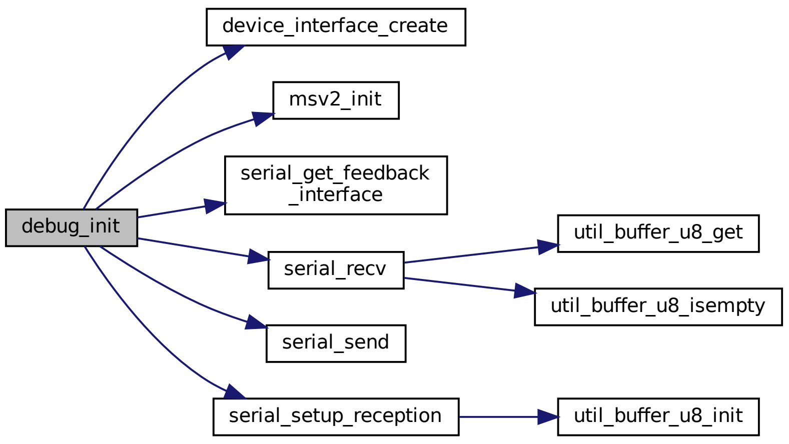
4.9.2 Function Documentation

4.9.2.1 debug_init()

```
util_error_t debug_init (
            void  )
```

References device_interface_create(), feedback_interface, feedback_interface_context, debug_interface_context↩
::msv2, msv2_init(), serial_deamon, serial_get_feedback_interface(), serial_recv(), serial_send(), serial_setup_↩
reception(), and SERIAL_TRANSFER_IT.

Here is the call graph for this function:



4.9.3 Variable Documentation

4.9.3.1 debug_context

```
debug_context_t debug_context
```

4.9.3.2 debug_interface_context

```
debug_interface_context_t debug_interface_context
```

4.9.3.3 feedback_interface_context

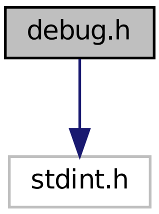
```
serial_interface_context_t feedback_interface_context
```

Referenced by debug_init().

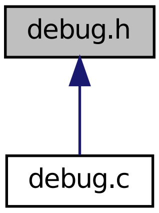
4.10 debug.h File Reference

```
#include <stdint.h>
```

Include dependency graph for debug.h:



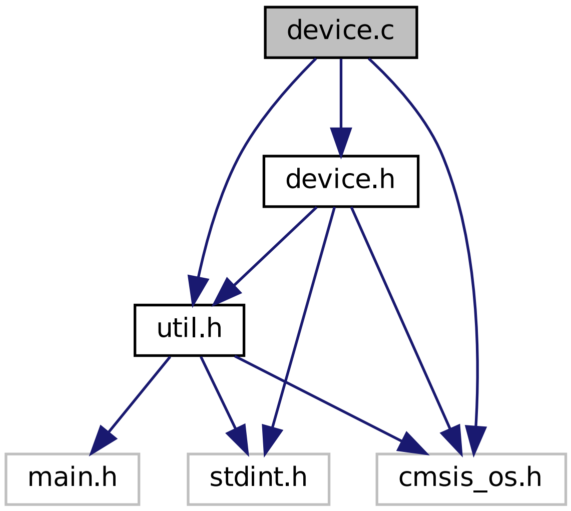
This graph shows which files directly or indirectly include this file:



4.11 device.c File Reference

```
#include <cmsis_os.h>
#include <util.h>
#include "device.h"
```

Include dependency graph for device.c:



Macros

- #define LEN_32 4
- #define LEN_16 2
- #define LEN_8 1

Functions

- void device_deamon_thread (void ∗arg)

  Generic device deamon thread.
- util_error_t device_create (device_t ∗dev, void ∗context, device_interface_t ∗interface, util_error_t(∗read_reg)(void ∗, device_interface_t ∗, uint32_t, uint8_t ∗, uint32_t), util_error_t(∗write_reg)(void ∗, device_interface_t ∗, uint32_t, uint8_t ∗, uint32_t))

  Initialize a device instance.
- util_error_t device_interface_create (device_interface_t ∗interface, void ∗context, device_deamon_t ∗deamon, util_error_t(∗send)(void ∗, uint8_t ∗, uint32_t), util_error_t(∗recv)(void ∗, uint8_t ∗, uint32_t ∗), util_error_t(∗handle_data)(void ∗, void ∗))

  Initialize a device interface instance.
- util_error_t device_deamon_create (device_deamon_t ∗deamon, const char ∗name, uint32_t prio, void ∗context, util_error_t(∗data_rdy)(void ∗))

  Initializea device deamon instance.
- util_error_t device_interface_send (device_interface_t ∗interface, uint8_t ∗data, uint32_t len)

  Send raw data through the interface specific send function.
- util_error_t device_interface_recv (device_interface_t ∗interface, uint8_t ∗data, uint32_t ∗len)

  Receive raw data through the interface specific recv function.
- util_error_t device_write_i32 (device_t ∗dev, uint32_t addr, int32_t data)

  Write an int32_t to a device register.
- util_error_t device_write_u32 (device_t ∗dev, uint32_t addr, uint32_t data)

  Write an uint32_t to a device register.
- util_error_t device_write_i16 (device_t ∗dev, uint32_t addr, int16_t data)

  Write an int16_t to a device register.
- util_error_t device_write_u16 (device_t ∗dev, uint32_t addr, uint16_t data)

  Write an uint16_t to a device register.
- util_error_t device_write_i8 (device_t ∗dev, uint32_t addr, int8_t data)

  Write an int8_t to a device register.
- util_error_t device_write_u8 (device_t ∗dev, uint32_t addr, uint8_t data)

  Write an uint8_t to a device register.
- util_error_t device_read_i32 (device_t ∗dev, uint32_t addr, int32_t ∗data)

  Read from an int32_t device register.
- util_error_t device_read_u32 (device_t ∗dev, uint32_t addr, uint32_t ∗data)

  Read from an uint32_t device register.
- util_error_t device_read_i16 (device_t ∗dev, uint32_t addr, int16_t ∗data)

  Read from an int16_t device register.
- util_error_t device_read_u16 (device_t ∗dev, uint32_t addr, uint16_t ∗data)

  Read from an uint16_t device register.
- util_error_t device_read_i8 (device_t ∗dev, uint32_t addr, int8_t ∗data)

  Read from an int8_t device register.
- util_error_t device_read_u8 (device_t ∗dev, uint32_t addr, uint8_t ∗data)

  Read from an uint8_t device register.

4.11.1 Macro Definition Documentation

4.11.1.1 LEN_16

```
#define LEN_16 2
```

4.11.1.2 LEN_32

```
#define LEN_32 4
```

4.11.1.3 LEN_8

```
#define LEN_8 1
```

4.11.2 Function Documentation

4.11.2.1 device_create()

```
util_error_t device_create (
             device_t * dev,
             void * context,
             device_interface_t * interface,
             util_error_t(*)(void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) read←
_reg,
             util_error_t(*)(void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) write←
_reg )
```

Initialize a device instance.

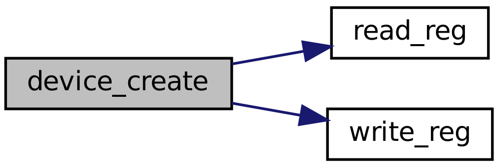
Parameters

dev	Pointer to the `device_t` structure describing this device.
context	Generic pointer to a device context.
interface	Pointer to the `device_interface_t` associated with this device.
read_reg	Pointer to a read register function for this device.
write_reg	Pointer to a write register function for this device.

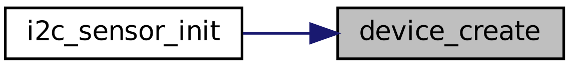
References device::context, ER_SUCCESS, device::id, device::interface, read_reg(), device::read_reg, write_reg(), and device::write_reg.

Referenced by i2c_sensor_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.2 device_deamon_create()

```
util_error_t device_deamon_create (
             device_deamon_t * deamon,
             const char * name,
             uint32_t prio,
             void * context,
             util_error_t(*)(void *) data_rdy )
```

Initializea device deamon instance.

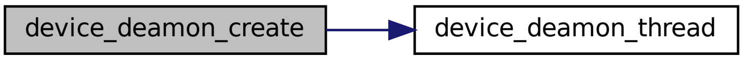
Parameters

deamon	
name	
prio	
context	
data_rdy	

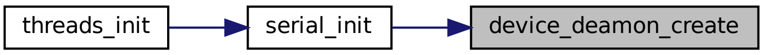
References device_deamon::buffer, device_deamon::context, device_deamon::data_rdy, DEAMON_STACK_↩ SIZE, device_deamon_thread(), ER_RESSOURCE_ERROR, ER_SUCCESS, device_deamon::handle, device_↩ deamon::id, device_deamon::interfaces_count, and device_deamon::stack.

Referenced by serial_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.3 device_deamon_thread()

```
void device_deamon_thread (
            void * arg )
```

Generic device deamon thread.

This thread will call the handle data function for an interface, whenever data is ready for a group of interfaces belonging to the same deamon.

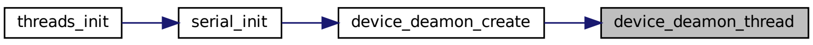
Parameters

arg	FreeRTOS entry point context, used to pass the deamon context to the thread.

References device_interface::context, device_deamon::context, device_deamon::data_rdy, ER_SUCCESS, device_interface::handle_data, and device_deamon::interfaces_count.

Referenced by device_deamon_create().

Here is the caller graph for this function:



4.11.2.4 device_interface_create()

```
util_error_t device_interface_create (
             device_interface_t * interface,
             void * context,
             device_deamon_t * deamon,
             util_error_t(*)(void *, uint8_t *, uint32_t) send,
             util_error_t(*)(void *, uint8_t *, uint32_t *) recv,
             util_error_t(*)(void *, void *) handle_data )
```

Initialize a device interface instance.

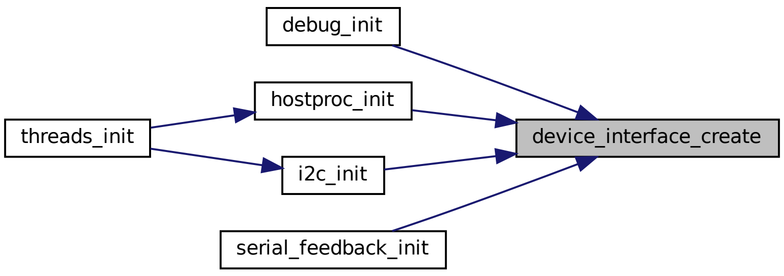
Parameters

interface	
context	
deamon	
send	
recv	
handle_data	

References device_interface::context, ER_SUCCESS, device_interface::handle_data, device_interface::id, device_deamon::interfaces, device_deamon::interfaces_count, device_interface::recv, and device_interface::send.

Referenced by debug_init(), hostproc_init(), i2c_init(), and serial_feedback_init().

Here is the caller graph for this function:



4.11.2.5 device_interface_recv()

```
util_error_t device_interface_recv (
             device_interface_t * interface,
             uint8_t * data,
             uint32_t * len )
```

Receive raw data through the interface specific recv function.

Parameters

interface	The interface through which data should be received.
data	A point to the data to be received.
len	A pointer to the length of the data to be received.

References device_interface::context, ER_RESSOURCE_ERROR, device_interface::recv, and device_interface↩
::send.

4.11.2.6 device_interface_send()

```
util_error_t device_interface_send (
             device_interface_t * interface,
             uint8_t * data,
             uint32_t len )
```

Send raw data through the interface specific send function.

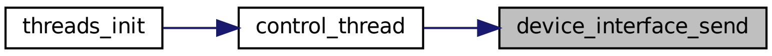
Parameters

interface	The interface through which data should be sent.
data	A point to the data to be sent.
len	The length of the data to be sent.

References device_interface::context, ER_RESSOURCE_ERROR, and device_interface::send.

Referenced by control_thread().

Here is the caller graph for this function:



4.11.2.7 device_read_i16()

```
util_error_t device_read_i16 (
             device_t * dev,
             uint32_t addr,
             int16_t * data )
```

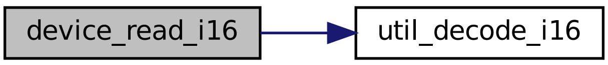
Read from an int16_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_i16().

Here is the call graph for this function:



4.11.2.8 device_read_i32()

```
util_error_t device_read_i32 (
             device_t * dev,
             uint32_t addr,
             int32_t * data )
```

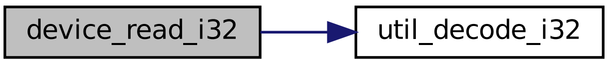
Read from an int32_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_32, device::read_reg, and util_decode_i32().

Here is the call graph for this function:



4.11.2.9 device_read_i8()

```
util_error_t device_read_i8 (
             device_t * dev,
             uint32_t addr,
             int8_t * data )
```

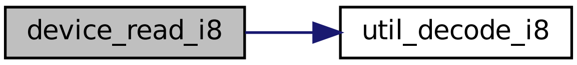
Read from an int8_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_8, device::read_reg, and util_decode_i8().

Here is the call graph for this function:



4.11.2.10 device_read_u16()

```
util_error_t device_read_u16 (
             device_t * dev,
             uint32_t addr,
             uint16_t * data )
```

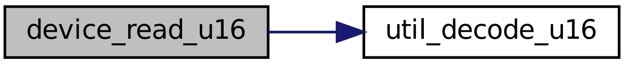
Read from an uint16_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_u16().

Here is the call graph for this function:



4.11.2.11 device_read_u32()

```
util_error_t device_read_u32 (
             device_t * dev,
             uint32_t addr,
             uint32_t * data )
```

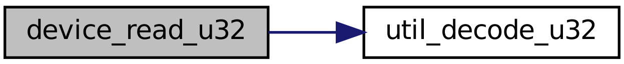
Read from an uint32_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_32, device::read_reg, and util_decode_u32().

Here is the call graph for this function:



4.11.2.12 device_read_u8()

```
util_error_t device_read_u8 (
             device_t * dev,
             uint32_t addr,
             uint8_t * data )
```

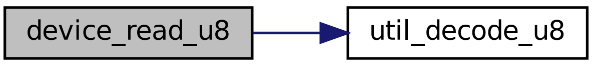
Read from an uint8_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_8, device::read_reg, and util_decode_u8().

Here is the call graph for this function:



4.11.2.13 device_write_i16()

```
util_error_t device_write_i16 (
             device_t * dev,
             uint32_t addr,
             int16_t data )
```

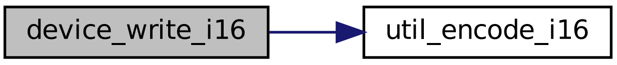
Write an int16_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_16, util_encode_i16(), and device::write_reg.

Here is the call graph for this function:



4.11.2.14 device_write_i32()

```
util_error_t device_write_i32 (
              device_t * dev,
              uint32_t addr,
              int32_t data )
```

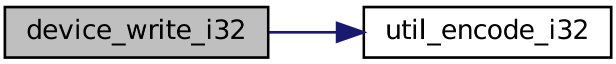
Write an int32_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_32, util_encode_i32(), and device::write_reg.

Here is the call graph for this function:



4.11.2.15 device_write_i8()

```
util_error_t device_write_i8 (
              device_t * dev,
              uint32_t addr,
              int8_t data )
```

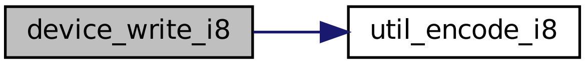
Write an int8_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_8, util_encode_i8(), and device::write_reg.

Here is the call graph for this function:



4.11.2.16 device_write_u16()

```
util_error_t device_write_u16 (
             device_t * dev,
             uint32_t addr,
             uint16_t data )
```

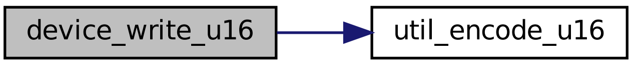
Write an uint16_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_16, util_encode_u16(), and device::write_reg.

Here is the call graph for this function:



4.11.2.17 device_write_u32()

```
util_error_t device_write_u32 (
             device_t * dev,
             uint32_t addr,
             uint32_t data )
```

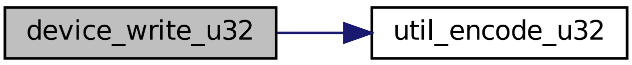
Write an uint32_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_32, util_encode_u32(), and device::write_reg.

Here is the call graph for this function:



4.11.2.18 device_write_u8()

```
util_error_t device_write_u8 (
              device_t * dev,
              uint32_t addr,
              uint8_t data )
```

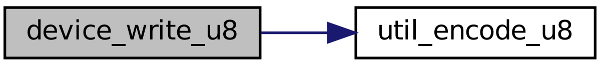
Write an uint8_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_8, util_encode_u8(), and device::write_reg.

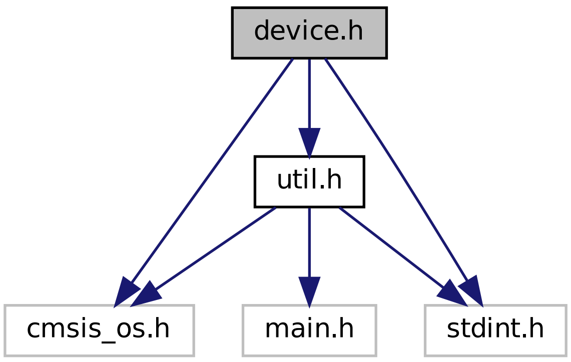
Here is the call graph for this function:



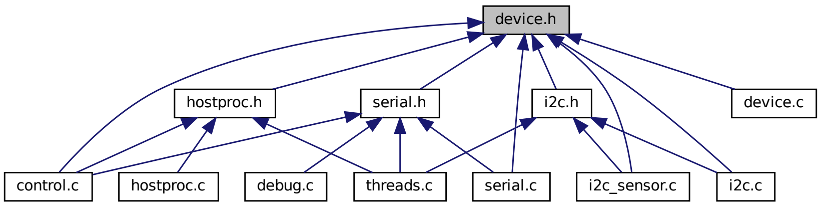
4.12 device.h File Reference

```
#include <cmsis_os.h>
#include <stdint.h>
#include <util.h>
```

Include dependency graph for device.h:



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



Data Structures

- struct device_interface
- struct device_deamon
- struct device

Macros

- #define DEVICE_NAME_LEN 16
- #define DEVICE_MAX_INTERFACES_PER_DEAMON 16
- #define DEAMON_STACK_SIZE 1024

Typedefs

- typedef struct device_interface device_interface_t
- typedef struct device_deamon device_deamon_t
- typedef struct device device_t

Functions

- util_error_t device_create (device_t ∗dev, void ∗context, device_interface_t ∗interface, util_error_t(∗read_reg)(void ∗, device_interface_t ∗, uint32_t, uint8_t ∗, uint32_t), util_error_t(∗write_reg)(void ∗, device_interface_t ∗, uint32_t, uint8_t ∗, uint32_t))

  Initialize a device instance.
- util_error_t device_deamon_create (device_deamon_t ∗deamon, const char ∗name, uint32_t prio, void ∗inst, util_error_t(∗data_rdy)(void ∗))

  Initializea device deamon instance.
- util_error_t device_interface_create (device_interface_t ∗interface, void ∗inst, device_deamon_t ∗deamon, util_error_t(∗send)(void ∗, uint8_t ∗, uint32_t), util_error_t(∗recv)(void ∗, uint8_t ∗, uint32_t ∗), util_error_t(∗handle_data)(void ∗, void ∗))

  Initialize a device interface instance.
- util_error_t device_interface_send (device_interface_t ∗interface, uint8_t ∗data, uint32_t len)

  Send raw data through the interface specific send function.
- util_error_t device_interface_recv (device_interface_t ∗interface, uint8_t ∗data, uint32_t ∗len)

  Receive raw data through the interface specific recv function.
- util_error_t device_write_i32 (device_t ∗dev, uint32_t addr, int32_t data)

  Write an int32_t to a device register.
- util_error_t device_write_u32 (device_t ∗dev, uint32_t addr, uint32_t data)

  Write an uint32_t to a device register.
- util_error_t device_write_i16 (device_t ∗dev, uint32_t addr, int16_t data)

  Write an int16_t to a device register.
- util_error_t device_write_u16 (device_t ∗dev, uint32_t addr, uint16_t data)

  Write an uint16_t to a device register.
- util_error_t device_write_i8 (device_t ∗dev, uint32_t addr, int8_t data)

  Write an int8_t to a device register.
- util_error_t device_write_u8 (device_t ∗dev, uint32_t addr, uint8_t data)

  Write an uint8_t to a device register.
- util_error_t device_read_i32 (device_t ∗dev, uint32_t addr, int32_t ∗data)

  Read from an int32_t device register.
- util_error_t device_read_u32 (device_t ∗dev, uint32_t addr, uint32_t ∗data)

  Read from an uint32_t device register.
- util_error_t device_read_i16 (device_t ∗dev, uint32_t addr, int16_t ∗data)

  Read from an int16_t device register.
- util_error_t device_read_u16 (device_t ∗dev, uint32_t addr, uint16_t ∗data)

  Read from an uint16_t device register.
- util_error_t device_read_i8 (device_t ∗dev, uint32_t addr, int8_t ∗data)

  Read from an int8_t device register.
- util_error_t device_read_u8 (device_t ∗dev, uint32_t addr, uint8_t ∗data)

  Read from an uint8_t device register.

4.12.1 Macro Definition Documentation

4.12.1.1 DEAMON_STACK_SIZE

```
#define DEAMON_STACK_SIZE 1024
```

4.12.1.2 DEVICE_MAX_INTERFACES_PER_DEAMON

```
#define DEVICE_MAX_INTERFACES_PER_DEAMON 16
```

4.12.1.3 DEVICE_NAME_LEN

```
#define DEVICE_NAME_LEN 16
```

4.12.2 Typedef Documentation

4.12.2.1 device_deamon_t

```
typedef struct device_deamon device_deamon_t
```

4.12.2.2 device_interface_t

```
typedef struct device_interface device_interface_t
```

4.12.2.3 device_t

```
typedef struct device device_t
```

4.12.3 Function Documentation

4.12.3.1 device_create()

```
util_error_t device_create (
             device_t * dev,
             void * context,
             device_interface_t * interface,
             util_error_t(*)(void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) read↩
_reg,
             util_error_t(*)(void *, device_interface_t *, uint32_t, uint8_t *, uint32_t) write↩
_reg )
```

Initialize a device instance.

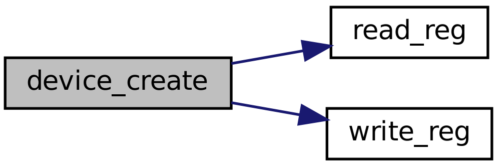
Parameters

dev	Pointer to the `device_t` structure describing this device.
context	Generic pointer to a device context.
interface	Pointer to the `device_interface_t` associated with this device.
read_reg	Pointer to a read register function for this device.
write_reg	Pointer to a write register function for this device.

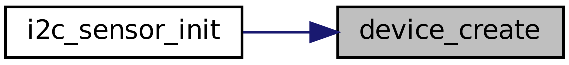
References device::context, ER_SUCCESS, device::id, device::interface, read_reg(), device::read_reg, write_reg(), and device::write_reg.

Referenced by i2c_sensor_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.3.2 device_deamon_create()

```
util_error_t device_deamon_create (
             device_deamon_t * deamon,
             const char * name,
             uint32_t prio,
             void * context,
             util_error_t(*)(void *) data_rdy )
```

Initializea device deamon instance.

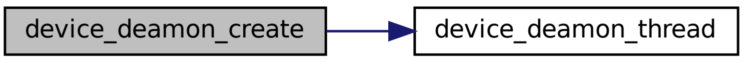
Parameters

deamon	
name	
prio	
context	
data_rdy	

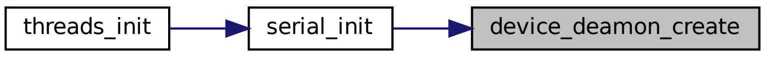
References device_deamon::buffer, device_deamon::context, device_deamon::data_rdy, DEAMON_STACK_↩ SIZE, device_deamon_thread(), ER_RESSOURCE_ERROR, ER_SUCCESS, device_deamon::handle, device_↩ deamon::id, device_deamon::interfaces_count, and device_deamon::stack.

Referenced by serial_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.3.3 device_interface_create()

```
util_error_t device_interface_create (
             device_interface_t * interface,
             void * context,
             device_deamon_t * deamon,
             util_error_t(*)(void *, uint8_t *, uint32_t) send,
             util_error_t(*)(void *, uint8_t *, uint32_t *) recv,
             util_error_t(*)(void *, void *) handle_data )
```

Initialize a device interface instance.

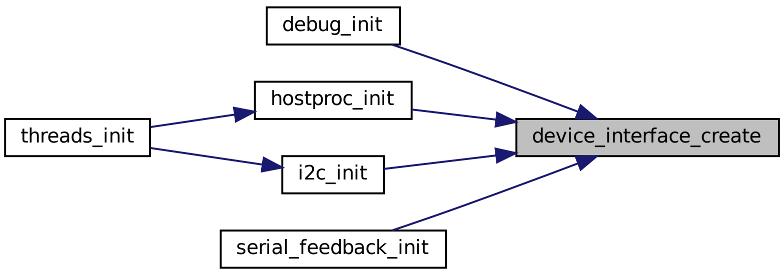
Parameters

interface	
context	
deamon	
send	
recv	
handle_data	

References device_interface::context, ER_SUCCESS, device_interface::handle_data, device_interface::id, device_deamon::interfaces, device_deamon::interfaces_count, device_interface::recv, and device_interface::send.

Referenced by debug_init(), hostproc_init(), i2c_init(), and serial_feedback_init().

Here is the caller graph for this function:



4.12.3.4 device_interface_recv()

```
util_error_t device_interface_recv (
             device_interface_t * interface,
             uint8_t * data,
             uint32_t * len )
```

Receive raw data through the interface specific recv function.

Parameters

interface	The interface through which data should be received.
data	A point to the data to be received.
len	A pointer to the length of the data to be received.

References device_interface::context, ER_RESSOURCE_ERROR, device_interface::recv, and device_interface↩ ::send.

4.12.3.5 device_interface_send()

```
util_error_t device_interface_send (
             device_interface_t * interface,
             uint8_t * data,
             uint32_t len )
```

Send raw data through the interface specific send function.

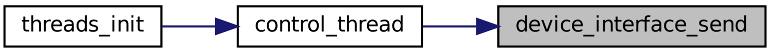
Parameters

interface	The interface through which data should be sent.
data	A point to the data to be sent.
len	The length of the data to be sent.

References device_interface::context, ER_RESSOURCE_ERROR, and device_interface::send.

Referenced by control_thread().

Here is the caller graph for this function:



4.12.3.6 device_read_i16()

```
util_error_t device_read_i16 (
             device_t * dev,
             uint32_t addr,
             int16_t * data )
```

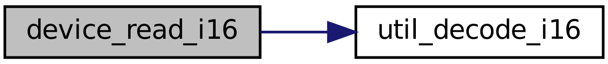
Read from an int16_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_i16().

Here is the call graph for this function:



4.12.3.7 device_read_i32()

```
util_error_t device_read_i32 (
             device_t * dev,
             uint32_t addr,
             int32_t * data )
```

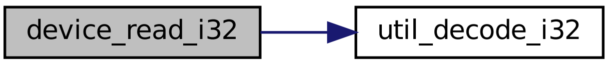
Read from an int32_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_32, device::read_reg, and util_decode_i32().

Here is the call graph for this function:



4.12.3.8 device_read_i8()

```
util_error_t device_read_i8 (
             device_t * dev,
             uint32_t addr,
             int8_t * data )
```

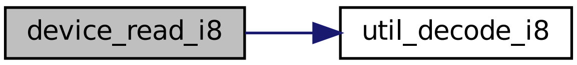
Read from an int8_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_8, device::read_reg, and util_decode_i8().

Here is the call graph for this function:



4.12.3.9 device_read_u16()

```
util_error_t device_read_u16 (
             device_t * dev,
             uint32_t addr,
             uint16_t * data )
```

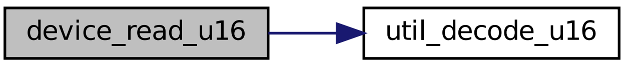
Read from an uint16_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_16, device::read_reg, and util_decode_u16().

Here is the call graph for this function:



4.12.3.10 device_read_u32()

```
util_error_t device_read_u32 (
             device_t * dev,
             uint32_t addr,
             uint32_t * data )
```

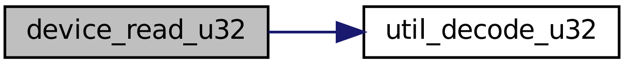
Read from an uint32_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_32, device::read_reg, and util_decode_u32().

Here is the call graph for this function:



4.12.3.11 device_read_u8()

```
util_error_t device_read_u8 (
             device_t * dev,
             uint32_t addr,
             uint8_t * data )
```

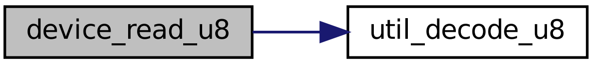
Read from an uint8_t device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	A pointer to the data to be read.

References device::context, ER_SUCCESS, device::interface, LEN_8, device::read_reg, and util_decode_u8().

Here is the call graph for this function:



4.12.3.12 device_write_i16()

```
util_error_t device_write_i16 (
             device_t * dev,
             uint32_t addr,
             int16_t data )
```

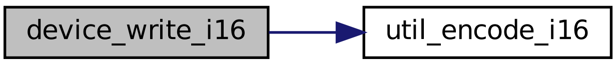
Write an int16_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_16, util_encode_i16(), and device::write_reg.

Here is the call graph for this function:



4.12.3.13 device_write_i32()

```
util_error_t device_write_i32 (
             device_t * dev,
             uint32_t addr,
             int32_t data )
```

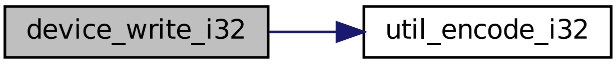
Write an int32_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_32, util_encode_i32(), and device::write_reg.

Here is the call graph for this function:



4.12.3.14 device_write_i8()

```
util_error_t device_write_i8 (
            device_t * dev,
            uint32_t addr,
            int8_t data )
```

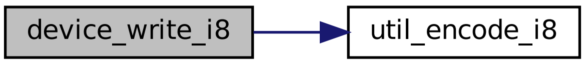
Write an int8_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_8, util_encode_i8(), and device::write_reg.

Here is the call graph for this function:



4.12.3.15 device_write_u16()

```
util_error_t device_write_u16 (
              device_t * dev,
              uint32_t addr,
              uint16_t data )
```

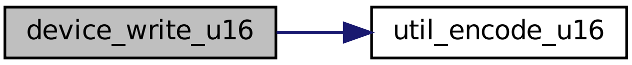
Write an uint16_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_16, util_encode_u16(), and device::write_reg.

Here is the call graph for this function:



4.12.3.16 device_write_u32()

```
util_error_t device_write_u32 (
              device_t * dev,
              uint32_t addr,
              uint32_t data )
```

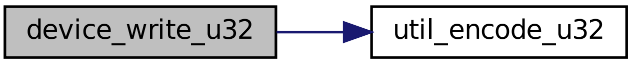
Write an uint32_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_32, util_encode_u32(), and device::write_reg.

Here is the call graph for this function:



4.12.3.17 device_write_u8()

```
util_error_t device_write_u8 (
             device_t * dev,
             uint32_t addr,
             uint8_t data )
```

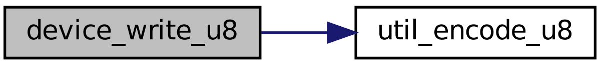
Write an uint8_t to a device register.

Parameters

dev	A pointer to the desired device.
addr	The address of the register.
data	The data to be written.

References device::context, ER_SUCCESS, device::interface, LEN_8, util_encode_u8(), and device::write_reg.

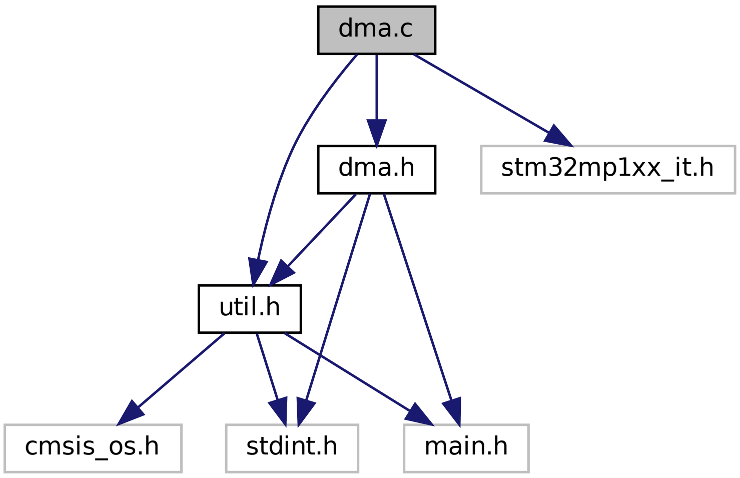
Here is the call graph for this function:



4.13 dma.c File Reference

```
#include "dma.h"
#include <util.h>
```

```
#include "stm32mp1xx_it.h"
```
Include dependency graph for dma.c:



Functions

- void dma_handle_interrupt (dma_stream_dev_t ∗stream)
- util_error_t dma2_init_scheduler (void)
- dma_scheduler_dev_t ∗ dma2_get_scheduler (void)
- dma_stream_dev_t ∗ dma2_get_streams (void)
- util_error_t dma_scheduler_init (dma_scheduler_dev_t ∗dma_scheduler, dma_stream_dev_t ∗dma_streams, uint16_t nb_dma_streams)
- dma_stream_dev_t ∗ dma_scheduler_request_stream (dma_scheduler_dev_t ∗dma_scheduler)
- util_error_t dma_scheduler_release_stream (dma_scheduler_dev_t ∗dma_scheduler, dma_stream_dev_t ∗dma_stream)
- util_error_t dma_start_stream (dma_stream_dev_t ∗stream, dma_stream_config_t config)

Variables

- dma_scheduler_dev_t dma2_scheduler
- dma_stream_dev_t dma2_streams [ ]

4.13.1 Function Documentation

4.13.1.1 dma2_get_scheduler()

```
dma_scheduler_dev_t* dma2_get_scheduler (
            void )
```

References dma2_scheduler.

4.13.1.2 dma2_get_streams()

```
dma_stream_dev_t* dma2_get_streams (
             void  )
```

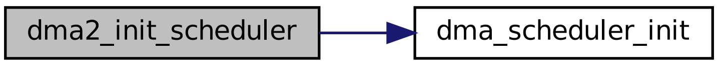
References dma2_streams.

4.13.1.3 dma2_init_scheduler()

```
util_error_t dma2_init_scheduler (
             void  )
```

References dma2_scheduler, dma2_streams, dma_scheduler_init(), and ER_SUCCESS.

Here is the call graph for this function:



4.13.1.4 dma_handle_interrupt()

```
void dma_handle_interrupt (
             dma_stream_dev_t * stream )
```

References dma_stream_dev::dma, DMA_STATUS_TC, DMA_STATUS_TE, DMA_STATUS_TH, dma_stream←_dev::number, dma_stream_dev::transfer_cplt, dma_stream_dev::transfer_error, dma_stream_dev::transfer_half, and dma_stream_dev::user_context.

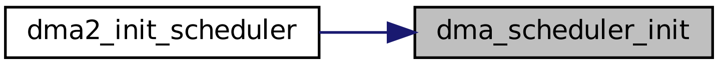
4.13.1.5 dma_scheduler_init()

```
util_error_t dma_scheduler_init (
             dma_scheduler_dev_t * dma_scheduler,
             dma_stream_dev_t * dma_streams,
             uint16_t nb_dma_streams )
```

References DMA_STREAM_FREE, DMA_STREAMS_MAX_LEN, ER_OUT_OF_RANGE, ER_SUCCESS, dma↩_stream_dev::state, and dma_scheduler_dev::streams.

Referenced by dma2_init_scheduler().

Here is the caller graph for this function:



4.13.1.6 dma_scheduler_release_stream()

```
util_error_t dma_scheduler_release_stream (
             dma_scheduler_dev_t * dma_scheduler,
             dma_stream_dev_t * dma_stream )
```

References DMA_STREAM_FREE, ENTER_CRITICAL, ER_SUCCESS, EXIT_CRITICAL, dma_scheduler_dev↩::free_stream_count, and dma_stream_dev::state.

4.13.1.7 dma_scheduler_request_stream()

```
dma_stream_dev_t* dma_scheduler_request_stream (
             dma_scheduler_dev_t * dma_scheduler )
```

References DMA_STREAM_BUSY, DMA_STREAM_FREE, ENTER_CRITICAL, EXIT_CRITICAL, dma_↩scheduler_dev::free_stream_count, dma_stream_dev::state, dma_scheduler_dev::stream_count, and dma_↩scheduler_dev::streams.

4.13.1.8 dma_start_stream()

```
util_error_t dma_start_stream (
             dma_stream_dev_t * stream,
             dma_stream_config_t config )
```

References dma_stream_config::direction, dma_stream_dev::dma, dma_stream_dev::dma_stream, dma_stream↩_dev::dmamux_channel, dma_stream_config::dmamux_request_number, ER_OUT_OF_RANGE, dma_stream↩_config::m0_addr, dma_stream_config::m1_addr, dma_stream_dev::number, dma_stream_config::p_addr, dma↩_stream_config::peripheral_flow_control, dma_stream_config::priority, dma_stream_config::transfer_cplt, dma↩_stream_dev::transfer_cplt, dma_stream_config::transfer_error, dma_stream_dev::transfer_error, dma_stream↩_config::transfer_half, dma_stream_dev::transfer_half, dma_stream_config::transfer_size, dma_stream_config↩::user_context, dma_stream_dev::user_context, and WRITE_IN_REG.

4.13.2 Variable Documentation

4.13.2.1 dma2_scheduler

```
dma_scheduler_dev_t dma2_scheduler
```

Referenced by dma2_get_scheduler(), and dma2_init_scheduler().

4.13.2.2 dma2_streams

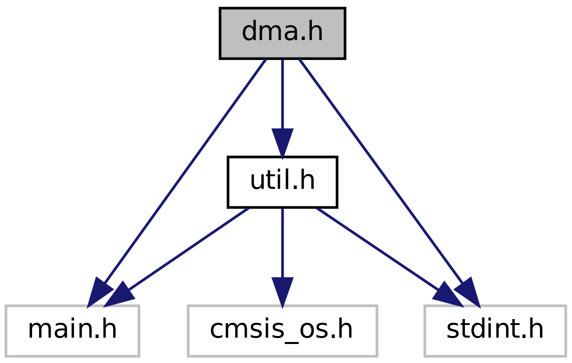
```
dma_stream_dev_t dma2_streams[]
```

Referenced by dma2_get_streams(), and dma2_init_scheduler().

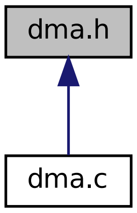
4.14 dma.h File Reference

```
#include <main.h>
#include <stdint.h>
#include <util.h>
```

Include dependency graph for dma.h:



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



Data Structures

- struct dma_request
- struct dma_stream_config
- struct dma_stream_dev
- struct dma_scheduler_dev

Macros

- #define DMA_STREAMS_MAX_LEN 8
- #define STM32_DMAMUX1_REQ_GEN0 1
- #define STM32_DMAMUX1_REQ_GEN1 2
- #define STM32_DMAMUX1_REQ_GEN2 3
- #define STM32_DMAMUX1_REQ_GEN3 4
- #define STM32_DMAMUX1_REQ_GEN4 5
- #define STM32_DMAMUX1_REQ_GEN5 6
- #define STM32_DMAMUX1_REQ_GEN6 7
- #define STM32_DMAMUX1_REQ_GEN7 8
- #define STM32_DMAMUX1_ADC1 9
- #define STM32_DMAMUX1_ADC2 10
- #define STM32_DMAMUX1_TIM1_CH1 11
- #define STM32_DMAMUX1_TIM1_CH2 12
- #define STM32_DMAMUX1_TIM1_CH3 13
- #define STM32_DMAMUX1_TIM1_CH4 14
- #define STM32_DMAMUX1_TIM1_UP 15
- #define STM32_DMAMUX1_TIM1_TRIG 16
- #define STM32_DMAMUX1_TIM1_COM 17
- #define STM32_DMAMUX1_TIM2_CH1 18
- #define STM32_DMAMUX1_TIM2_CH2 19
- #define STM32_DMAMUX1_TIM2_CH3 20
- #define STM32_DMAMUX1_TIM2_CH4 21
- #define STM32_DMAMUX1_TIM2_UP 22
- #define STM32_DMAMUX1_TIM3_CH1 23
- #define STM32_DMAMUX1_TIM3_CH2 24
- #define STM32_DMAMUX1_TIM3_CH3 25
- #define STM32_DMAMUX1_TIM3_CH4 26

- #define STM32_DMAMUX1_TIM3_UP 27
- #define STM32_DMAMUX1_TIM3_TRIG 28
- #define STM32_DMAMUX1_TIM4_CH1 29
- #define STM32_DMAMUX1_TIM4_CH2 30
- #define STM32_DMAMUX1_TIM4_CH3 31
- #define STM32_DMAMUX1_TIM4_UP 32
- #define STM32_DMAMUX1_I2C1_RX 33
- #define STM32_DMAMUX1_I2C1_TX 34
- #define STM32_DMAMUX1_I2C2_RX 35
- #define STM32_DMAMUX1_I2C2_TX 36
- #define STM32_DMAMUX1_SPI1_RX 37
- #define STM32_DMAMUX1_SPI1_TX 38
- #define STM32_DMAMUX1_SPI2_RX 39
- #define STM32_DMAMUX1_SPI2_TX 40
- #define STM32_DMAMUX1_RSVD41 41
- #define STM32_DMAMUX1_RSVD42 42
- #define STM32_DMAMUX1_USART2_RX 43
- #define STM32_DMAMUX1_USART2_TX 44
- #define STM32_DMAMUX1_USART3_RX 45
- #define STM32_DMAMUX1_USART3_TX 46
- #define STM32_DMAMUX1_TIM8_CH1 47
- #define STM32_DMAMUX1_TIM8_CH2 48
- #define STM32_DMAMUX1_TIM8_CH3 49
- #define STM32_DMAMUX1_TIM8_CH4 50
- #define STM32_DMAMUX1_TIM8_UP 51
- #define STM32_DMAMUX1_TIM8_TRIG 52
- #define STM32_DMAMUX1_TIM8_COM 53
- #define STM32_DMAMUX1_RSVD54 54
- #define STM32_DMAMUX1_TIM5_CH1 55
- #define STM32_DMAMUX1_TIM5_CH2 56
- #define STM32_DMAMUX1_TIM5_CH3 57
- #define STM32_DMAMUX1_TIM5_CH4 58
- #define STM32_DMAMUX1_TIM5_UP 59
- #define STM32_DMAMUX1_TIM5_TRIG 60
- #define STM32_DMAMUX1_SPI3_RX 61
- #define STM32_DMAMUX1_SPI3_TX 62
- #define STM32_DMAMUX1_UART4_RX 63
- #define STM32_DMAMUX1_UART4_TX 64
- #define STM32_DMAMUX1_UART5_RX 65
- #define STM32_DMAMUX1_UART5_TX 66
- #define STM32_DMAMUX1_DAC1_CH1 67
- #define STM32_DMAMUX1_DAC1_CH2 68
- #define STM32_DMAMUX1_TIM6_UP 69
- #define STM32_DMAMUX1_TIM7_UP 70
- #define STM32_DMAMUX1_USART6_RX 71
- #define STM32_DMAMUX1_USART6_TX 72
- #define STM32_DMAMUX1_I2C3_RX 73
- #define STM32_DMAMUX1_I2C3_TX 74
- #define STM32_DMAMUX1_DCMI 75
- #define STM32_DMAMUX1_CRYP2_IN 76
- #define STM32_DMAMUX1_CRYP2_OUT 77
- #define STM32_DMAMUX1_HASH2_IN 78
- #define STM32_DMAMUX1_UART7_RX 79
- #define STM32_DMAMUX1_UART7_TX 80
- #define STM32_DMAMUX1_UART8_RX 81

- #define STM32_DMAMUX1_UART8_TX 82
- #define STM32_DMAMUX1_SPI4_RX 83
- #define STM32_DMAMUX1_SPI4_TX 84
- #define STM32_DMAMUX1_SPI5_RX 85
- #define STM32_DMAMUX1_SPI5_TX 86
- #define STM32_DMAMUX1_SAI1_A 87
- #define STM32_DMAMUX1_SAI1_B 88
- #define STM32_DMAMUX1_SAI2_A 89
- #define STM32_DMAMUX1_SAI2_B 90
- #define STM32_DMAMUX1_DFSDM1_FLT4 91
- #define STM32_DMAMUX1_DFSDM1_FLT5 92
- #define STM32_DMAMUX1_SPDIFRX_DT 93
- #define STM32_DMAMUX1_SPDIFRX_CS 94
- #define STM32_DMAMUX1_RSVD95 95
- #define STM32_DMAMUX1_RSVD96 96
- #define STM32_DMAMUX1_RSVD97 97
- #define STM32_DMAMUX1_RSVD98 98
- #define STM32_DMAMUX1_SAI4_A 99
- #define STM32_DMAMUX1_SAI4_B 100
- #define STM32_DMAMUX1_DFSDM1_FLT0 101
- #define STM32_DMAMUX1_DFSDM1_FLT1 102
- #define STM32_DMAMUX1_DFSDM1_FLT2 103
- #define STM32_DMAMUX1_DFSDM1_FLT3 104
- #define STM32_DMAMUX1_TIM15_CH1 105
- #define STM32_DMAMUX1_TIM15_UP 106
- #define STM32_DMAMUX1_TIM15_TRIG 107
- #define STM32_DMAMUX1_TIM15_COM 108
- #define STM32_DMAMUX1_TIM16_CH1 109
- #define STM32_DMAMUX1_TIM16_UP 110
- #define STM32_DMAMUX1_TIM17_CH1 111
- #define STM32_DMAMUX1_TIM17_UP 112
- #define STM32_DMAMUX1_SAI3_A 113
- #define STM32_DMAMUX1_SAI3_B 114
- #define STM32_DMAMUX1_I2C5_RX 115
- #define STM32_DMAMUX1_I2C5_TX 116
- #define STM32_DMAMUX1_RSVD117 117
- #define STM32_DMAMUX1_RSVD118 118
- #define STM32_DMAMUX1_RSVD119 119
- #define STM32_DMAMUX1_RSVD120 120
- #define STM32_DMAMUX1_RSVD121 121
- #define STM32_DMAMUX1_RSVD122 122
- #define STM32_DMAMUX1_RSVD123 123
- #define STM32_DMAMUX1_RSVD124 124
- #define STM32_DMAMUX1_RSVD125 125
- #define STM32_DMAMUX1_RSVD126 126
- #define STM32_DMAMUX1_RSVD127 127
- #define DMA_STATUS_TC (0b1<<5)
- #define DMA_STATUS_TH (0b1<<4)
- #define DMA_STATUS_TE (0b1<<3)

Typedefs

- typedef enum dma_stream_state dma_stream_state_t
- typedef enum dma_stream_dir dma_stream_dir_t
- typedef struct dma_request dma_request_t
- typedef struct dma_stream_config dma_stream_config_t
- typedef struct dma_stream_dev dma_stream_dev_t
- typedef struct dma_scheduler_dev dma_scheduler_dev_t

Enumerations

- enum dma_stream_state { DMA_STREAM_BUSY , DMA_STREAM_FREE }
- enum dma_stream_dir { DMA_STREAM_P2M = 0b00 , DMA_STREAM_M2P = 0b01 , DMA_STREAM_M2M = 0b10 }

Functions

- util_error_t dma2_init_scheduler (void)
- dma_scheduler_dev_t ∗ dma2_get_scheduler (void)
- dma_stream_dev_t ∗ dma2_get_streams (void)
- util_error_t dma_scheduler_init (dma_scheduler_dev_t ∗dma_scheduler, dma_stream_dev_t ∗dma_streams, uint16_t nb_dma_streams)
- dma_stream_dev_t ∗ dma_scheduler_request_stream (dma_scheduler_dev_t ∗dma_scheduler)
- util_error_t dma_scheduler_release_stream (dma_scheduler_dev_t ∗dma_scheduler, dma_stream_dev_t ∗dma_stream)
- util_error_t dma_start_stream (dma_stream_dev_t ∗stream, dma_stream_config_t config)
- util_error_t dma_stop_stream (dma_stream_dev_t ∗stream)
- util_error_t dma_copy (void ∗dst, void ∗src, uint32_t len)

4.14.1 Macro Definition Documentation

4.14.1.1 DMA_STATUS_TC

```
#define DMA_STATUS_TC (0b1<<5)
```

4.14.1.2 DMA_STATUS_TE

```
#define DMA_STATUS_TE (0b1<<3)
```

4.14.1.3 DMA_STATUS_TH

```
#define DMA_STATUS_TH (0b1<<4)
```

4.14.1.4 DMA_STREAMS_MAX_LEN

```
#define DMA_STREAMS_MAX_LEN 8
```

4.14.1.5 STM32_DMAMUX1_ADC1

```
#define STM32_DMAMUX1_ADC1 9
```

4.14.1.6 STM32_DMAMUX1_ADC2

```
#define STM32_DMAMUX1_ADC2 10
```

4.14.1.7 STM32_DMAMUX1_CRYP2_IN

```
#define STM32_DMAMUX1_CRYP2_IN 76
```

4.14.1.8 STM32_DMAMUX1_CRYP2_OUT

```
#define STM32_DMAMUX1_CRYP2_OUT 77
```

4.14.1.9 STM32_DMAMUX1_DAC1_CH1

```
#define STM32_DMAMUX1_DAC1_CH1 67
```

4.14.1.10 STM32_DMAMUX1_DAC1_CH2

```
#define STM32_DMAMUX1_DAC1_CH2 68
```

4.14.1.11 STM32_DMAMUX1_DCMI

```
#define STM32_DMAMUX1_DCMI 75
```

4.14.1.12 STM32_DMAMUX1_DFSDM1_FLT0

```
#define STM32_DMAMUX1_DFSDM1_FLT0 101
```

4.14.1.13 STM32_DMAMUX1_DFSDM1_FLT1

```
#define STM32_DMAMUX1_DFSDM1_FLT1 102
```

4.14.1.14 STM32_DMAMUX1_DFSDM1_FLT2

```
#define STM32_DMAMUX1_DFSDM1_FLT2 103
```

4.14.1.15 STM32_DMAMUX1_DFSDM1_FLT3

```
#define STM32_DMAMUX1_DFSDM1_FLT3 104
```

4.14.1.16 STM32_DMAMUX1_DFSDM1_FLT4

```
#define STM32_DMAMUX1_DFSDM1_FLT4 91
```

4.14.1.17 STM32_DMAMUX1_DFSDM1_FLT5

```
#define STM32_DMAMUX1_DFSDM1_FLT5 92
```

4.14.1.18 STM32_DMAMUX1_HASH2_IN

```
#define STM32_DMAMUX1_HASH2_IN 78
```

4.14.1.19 STM32_DMAMUX1_I2C1_RX

```
#define STM32_DMAMUX1_I2C1_RX 33
```

4.14.1.20 STM32_DMAMUX1_I2C1_TX

```
#define STM32_DMAMUX1_I2C1_TX 34
```

4.14.1.21 STM32_DMAMUX1_I2C2_RX

```
#define STM32_DMAMUX1_I2C2_RX 35
```

4.14.1.22 STM32_DMAMUX1_I2C2_TX

```
#define STM32_DMAMUX1_I2C2_TX 36
```

4.14.1.23 STM32_DMAMUX1_I2C3_RX

```
#define STM32_DMAMUX1_I2C3_RX 73
```

4.14.1.24 STM32_DMAMUX1_I2C3_TX

```
#define STM32_DMAMUX1_I2C3_TX 74
```

4.14.1.25 STM32_DMAMUX1_I2C5_RX

```
#define STM32_DMAMUX1_I2C5_RX 115
```

4.14.1.26 STM32_DMAMUX1_I2C5_TX

```
#define STM32_DMAMUX1_I2C5_TX 116
```

4.14.1.27 STM32_DMAMUX1_REQ_GEN0

```
#define STM32_DMAMUX1_REQ_GEN0 1
```

4.14.1.28 STM32_DMAMUX1_REQ_GEN1

```
#define STM32_DMAMUX1_REQ_GEN1 2
```

4.14.1.29 STM32_DMAMUX1_REQ_GEN2

```
#define STM32_DMAMUX1_REQ_GEN2 3
```

4.14.1.30 STM32_DMAMUX1_REQ_GEN3

```
#define STM32_DMAMUX1_REQ_GEN3 4
```

4.14.1.31 STM32_DMAMUX1_REQ_GEN4

```
#define STM32_DMAMUX1_REQ_GEN4 5
```

4.14.1.32 STM32_DMAMUX1_REQ_GEN5

```
#define STM32_DMAMUX1_REQ_GEN5 6
```

4.14.1.33 STM32_DMAMUX1_REQ_GEN6

```
#define STM32_DMAMUX1_REQ_GEN6 7
```

4.14.1.34 STM32_DMAMUX1_REQ_GEN7

```
#define STM32_DMAMUX1_REQ_GEN7 8
```

4.14.1.35 STM32_DMAMUX1_RSVD117

```
#define STM32_DMAMUX1_RSVD117 117
```

4.14.1.36 STM32_DMAMUX1_RSVD118

```
#define STM32_DMAMUX1_RSVD118 118
```

4.14.1.37 STM32_DMAMUX1_RSVD119

```
#define STM32_DMAMUX1_RSVD119 119
```

4.14.1.38 STM32_DMAMUX1_RSVD120

```
#define STM32_DMAMUX1_RSVD120 120
```

4.14.1.39 STM32_DMAMUX1_RSVD121

```
#define STM32_DMAMUX1_RSVD121 121
```

4.14.1.40 STM32_DMAMUX1_RSVD122

```
#define STM32_DMAMUX1_RSVD122 122
```

4.14.1.41 STM32_DMAMUX1_RSVD123

```
#define STM32_DMAMUX1_RSVD123 123
```

4.14.1.42 STM32_DMAMUX1_RSVD124

```
#define STM32_DMAMUX1_RSVD124 124
```

4.14.1.43 STM32_DMAMUX1_RSVD125

```
#define STM32_DMAMUX1_RSVD125 125
```

4.14.1.44 STM32_DMAMUX1_RSVD126

```
#define STM32_DMAMUX1_RSVD126 126
```

4.14.1.45 STM32_DMAMUX1_RSVD127

```
#define STM32_DMAMUX1_RSVD127 127
```

4.14.1.46 STM32_DMAMUX1_RSVD41

```
#define STM32_DMAMUX1_RSVD41 41
```

4.14.1.47 STM32_DMAMUX1_RSVD42

```
#define STM32_DMAMUX1_RSVD42 42
```

4.14.1.48 STM32_DMAMUX1_RSVD54

```
#define STM32_DMAMUX1_RSVD54 54
```

4.14.1.49 STM32_DMAMUX1_RSVD95

```
#define STM32_DMAMUX1_RSVD95 95
```

4.14.1.50 STM32_DMAMUX1_RSVD96

```
#define STM32_DMAMUX1_RSVD96 96
```

4.14.1.51 STM32_DMAMUX1_RSVD97

```
#define STM32_DMAMUX1_RSVD97 97
```

4.14.1.52 STM32_DMAMUX1_RSVD98

```
#define STM32_DMAMUX1_RSVD98 98
```

4.14.1.53 STM32_DMAMUX1_SAI1_A

```
#define STM32_DMAMUX1_SAI1_A 87
```

4.14.1.54 STM32_DMAMUX1_SAI1_B

```
#define STM32_DMAMUX1_SAI1_B 88
```

4.14.1.55 STM32_DMAMUX1_SAI2_A

```
#define STM32_DMAMUX1_SAI2_A 89
```

4.14.1.56 STM32_DMAMUX1_SAI2_B

```
#define STM32_DMAMUX1_SAI2_B 90
```

4.14.1.57 STM32_DMAMUX1_SAI3_A

```
#define STM32_DMAMUX1_SAI3_A 113
```

4.14.1.58 STM32_DMAMUX1_SAI3_B

```
#define STM32_DMAMUX1_SAI3_B 114
```

4.14.1.59 STM32_DMAMUX1_SAI4_A

```
#define STM32_DMAMUX1_SAI4_A 99
```

4.14.1.60 STM32_DMAMUX1_SAI4_B

```
#define STM32_DMAMUX1_SAI4_B 100
```

4.14.1.61 STM32_DMAMUX1_SPDIFRX_CS

```
#define STM32_DMAMUX1_SPDIFRX_CS 94
```

4.14.1.62 STM32_DMAMUX1_SPDIFRX_DT

```
#define STM32_DMAMUX1_SPDIFRX_DT 93
```

4.14.1.63 STM32_DMAMUX1_SPI1_RX

```
#define STM32_DMAMUX1_SPI1_RX 37
```

4.14.1.64 STM32_DMAMUX1_SPI1_TX

```
#define STM32_DMAMUX1_SPI1_TX 38
```

4.14.1.65 STM32_DMAMUX1_SPI2_RX

```
#define STM32_DMAMUX1_SPI2_RX 39
```

4.14.1.66 STM32_DMAMUX1_SPI2_TX

```
#define STM32_DMAMUX1_SPI2_TX 40
```

4.14.1.67 STM32_DMAMUX1_SPI3_RX

```
#define STM32_DMAMUX1_SPI3_RX 61
```

4.14.1.68 STM32_DMAMUX1_SPI3_TX

```
#define STM32_DMAMUX1_SPI3_TX 62
```

4.14.1.69 STM32_DMAMUX1_SPI4_RX

```
#define STM32_DMAMUX1_SPI4_RX 83
```

4.14.1.70 STM32_DMAMUX1_SPI4_TX

```
#define STM32_DMAMUX1_SPI4_TX 84
```

4.14.1.71 STM32_DMAMUX1_SPI5_RX

```
#define STM32_DMAMUX1_SPI5_RX 85
```

4.14.1.72 STM32_DMAMUX1_SPI5_TX

```
#define STM32_DMAMUX1_SPI5_TX 86
```

4.14.1.73 STM32_DMAMUX1_TIM15_CH1

```
#define STM32_DMAMUX1_TIM15_CH1 105
```

4.14.1.74 STM32_DMAMUX1_TIM15_COM

```
#define STM32_DMAMUX1_TIM15_COM 108
```

4.14.1.75 STM32_DMAMUX1_TIM15_TRIG

```
#define STM32_DMAMUX1_TIM15_TRIG 107
```

4.14.1.76 STM32_DMAMUX1_TIM15_UP

```
#define STM32_DMAMUX1_TIM15_UP 106
```

4.14.1.77 STM32_DMAMUX1_TIM16_CH1

```
#define STM32_DMAMUX1_TIM16_CH1 109
```

4.14.1.78 STM32_DMAMUX1_TIM16_UP

```
#define STM32_DMAMUX1_TIM16_UP 110
```

4.14.1.79 STM32_DMAMUX1_TIM17_CH1

```
#define STM32_DMAMUX1_TIM17_CH1 111
```

4.14.1.80 STM32_DMAMUX1_TIM17_UP

```
#define STM32_DMAMUX1_TIM17_UP 112
```

4.14.1.81 STM32_DMAMUX1_TIM1_CH1

```
#define STM32_DMAMUX1_TIM1_CH1 11
```

4.14.1.82 STM32_DMAMUX1_TIM1_CH2

```
#define STM32_DMAMUX1_TIM1_CH2 12
```

4.14.1.83 STM32_DMAMUX1_TIM1_CH3

```
#define STM32_DMAMUX1_TIM1_CH3 13
```

4.14.1.84 STM32_DMAMUX1_TIM1_CH4

```
#define STM32_DMAMUX1_TIM1_CH4 14
```

4.14.1.85 STM32_DMAMUX1_TIM1_COM

```
#define STM32_DMAMUX1_TIM1_COM 17
```

4.14.1.86 STM32_DMAMUX1_TIM1_TRIG

```
#define STM32_DMAMUX1_TIM1_TRIG 16
```

4.14.1.87 STM32_DMAMUX1_TIM1_UP

```
#define STM32_DMAMUX1_TIM1_UP 15
```

4.14.1.88 STM32_DMAMUX1_TIM2_CH1

```
#define STM32_DMAMUX1_TIM2_CH1 18
```

4.14.1.89 STM32_DMAMUX1_TIM2_CH2

```
#define STM32_DMAMUX1_TIM2_CH2 19
```

4.14.1.90 STM32_DMAMUX1_TIM2_CH3

```
#define STM32_DMAMUX1_TIM2_CH3 20
```

4.14.1.91 STM32_DMAMUX1_TIM2_CH4

```
#define STM32_DMAMUX1_TIM2_CH4 21
```

4.14.1.92 STM32_DMAMUX1_TIM2_UP

```
#define STM32_DMAMUX1_TIM2_UP 22
```

4.14.1.93 STM32_DMAMUX1_TIM3_CH1

```
#define STM32_DMAMUX1_TIM3_CH1 23
```

4.14.1.94 STM32_DMAMUX1_TIM3_CH2

```
#define STM32_DMAMUX1_TIM3_CH2 24
```

4.14.1.95 STM32_DMAMUX1_TIM3_CH3

```
#define STM32_DMAMUX1_TIM3_CH3 25
```

4.14.1.96 STM32_DMAMUX1_TIM3_CH4

```
#define STM32_DMAMUX1_TIM3_CH4 26
```

4.14.1.97 STM32_DMAMUX1_TIM3_TRIG

```
#define STM32_DMAMUX1_TIM3_TRIG 28
```

4.14.1.98 STM32_DMAMUX1_TIM3_UP

```
#define STM32_DMAMUX1_TIM3_UP 27
```

4.14.1.99 STM32_DMAMUX1_TIM4_CH1

```
#define STM32_DMAMUX1_TIM4_CH1 29
```

4.14.1.100 STM32_DMAMUX1_TIM4_CH2

```
#define STM32_DMAMUX1_TIM4_CH2 30
```

4.14.1.101 STM32_DMAMUX1_TIM4_CH3

```
#define STM32_DMAMUX1_TIM4_CH3 31
```

4.14.1.102 STM32_DMAMUX1_TIM4_UP

```
#define STM32_DMAMUX1_TIM4_UP 32
```

4.14.1.103 STM32_DMAMUX1_TIM5_CH1

```
#define STM32_DMAMUX1_TIM5_CH1 55
```

4.14.1.104 STM32_DMAMUX1_TIM5_CH2

```
#define STM32_DMAMUX1_TIM5_CH2 56
```

4.14.1.105 STM32_DMAMUX1_TIM5_CH3

```
#define STM32_DMAMUX1_TIM5_CH3 57
```

4.14.1.106 STM32_DMAMUX1_TIM5_CH4

```
#define STM32_DMAMUX1_TIM5_CH4 58
```

4.14.1.107 STM32_DMAMUX1_TIM5_TRIG

```
#define STM32_DMAMUX1_TIM5_TRIG 60
```

4.14.1.108 STM32_DMAMUX1_TIM5_UP

```
#define STM32_DMAMUX1_TIM5_UP 59
```

4.14.1.109 STM32_DMAMUX1_TIM6_UP

```
#define STM32_DMAMUX1_TIM6_UP 69
```

4.14.1.110 STM32_DMAMUX1_TIM7_UP

```
#define STM32_DMAMUX1_TIM7_UP 70
```

4.14.1.111 STM32_DMAMUX1_TIM8_CH1

```
#define STM32_DMAMUX1_TIM8_CH1 47
```

4.14.1.112 STM32_DMAMUX1_TIM8_CH2

```
#define STM32_DMAMUX1_TIM8_CH2 48
```

4.14.1.113 STM32_DMAMUX1_TIM8_CH3

```
#define STM32_DMAMUX1_TIM8_CH3 49
```

4.14.1.114 STM32_DMAMUX1_TIM8_CH4

```
#define STM32_DMAMUX1_TIM8_CH4 50
```

4.14.1.115 STM32_DMAMUX1_TIM8_COM

```
#define STM32_DMAMUX1_TIM8_COM 53
```

4.14.1.116 STM32_DMAMUX1_TIM8_TRIG

```
#define STM32_DMAMUX1_TIM8_TRIG 52
```

4.14.1.117 STM32_DMAMUX1_TIM8_UP

```
#define STM32_DMAMUX1_TIM8_UP 51
```

4.14.1.118 STM32_DMAMUX1_UART4_RX

```
#define STM32_DMAMUX1_UART4_RX 63
```

4.14.1.119 STM32_DMAMUX1_UART4_TX

```
#define STM32_DMAMUX1_UART4_TX 64
```

4.14.1.120 STM32_DMAMUX1_UART5_RX

```
#define STM32_DMAMUX1_UART5_RX 65
```

4.14.1.121 STM32_DMAMUX1_UART5_TX

```
#define STM32_DMAMUX1_UART5_TX 66
```

4.14.1.122 STM32_DMAMUX1_UART7_RX

```
#define STM32_DMAMUX1_UART7_RX 79
```

4.14.1.123 STM32_DMAMUX1_UART7_TX

```
#define STM32_DMAMUX1_UART7_TX 80
```

4.14.1.124 STM32_DMAMUX1_UART8_RX

```
#define STM32_DMAMUX1_UART8_RX 81
```

4.14.1.125 STM32_DMAMUX1_UART8_TX

```
#define STM32_DMAMUX1_UART8_TX 82
```

4.14.1.126 STM32_DMAMUX1_USART2_RX

```
#define STM32_DMAMUX1_USART2_RX 43
```

4.14.1.127 STM32_DMAMUX1_USART2_TX

```
#define STM32_DMAMUX1_USART2_TX 44
```

4.14.1.128 STM32_DMAMUX1_USART3_RX

```
#define STM32_DMAMUX1_USART3_RX 45
```

4.14.1.129 STM32_DMAMUX1_USART3_TX

```
#define STM32_DMAMUX1_USART3_TX 46
```

4.14.1.130 STM32_DMAMUX1_USART6_RX

```
#define STM32_DMAMUX1_USART6_RX 71
```

4.14.1.131 STM32_DMAMUX1_USART6_TX

```
#define STM32_DMAMUX1_USART6_TX 72
```

4.14.2 Typedef Documentation

4.14.2.1 dma_request_t

```
typedef struct dma_request dma_request_t
```

4.14.2.2 dma_scheduler_dev_t

```
typedef struct dma_scheduler_dev dma_scheduler_dev_t
```

4.14.2.3 dma_stream_config_t

```
typedef struct dma_stream_config dma_stream_config_t
```

4.14.2.4 dma_stream_dev_t

```
typedef struct dma_stream_dev dma_stream_dev_t
```

4.14.2.5 dma_stream_dir_t

```
typedef enum dma_stream_dir dma_stream_dir_t
```

4.14.2.6 dma_stream_state_t

```
typedef enum dma_stream_state dma_stream_state_t
```

4.14.3 Enumeration Type Documentation

4.14.3.1 dma_stream_dir

```
enum dma_stream_dir
```

Enumerator

DMA_STREAM_P2M	
DMA_STREAM_M2P	
DMA_STREAM_M2M	

4.14.3.2 dma_stream_state

```
enum dma_stream_state
```

Enumerator

DMA_STREAM_BUSY	
DMA_STREAM_FREE	

4.14.4 Function Documentation

4.14.4.1 dma2_get_scheduler()

```
dma_scheduler_dev_t* dma2_get_scheduler (
             void  )
```

References dma2_scheduler.

4.14.4.2 dma2_get_streams()

```
dma_stream_dev_t* dma2_get_streams (
             void  )
```

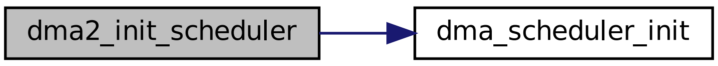
References dma2_streams.

4.14.4.3 dma2_init_scheduler()

```
util_error_t dma2_init_scheduler (
             void  )
```

References dma2_scheduler, dma2_streams, dma_scheduler_init(), and ER_SUCCESS.

Here is the call graph for this function:



4.14.4.4 dma_copy()

```
util_error_t dma_copy (
             void * dst,
             void * src,
             uint32_t len )
```

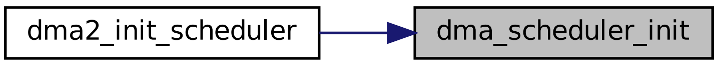
4.14.4.5 dma_scheduler_init()

```
util_error_t dma_scheduler_init (
             dma_scheduler_dev_t * dma_scheduler,
             dma_stream_dev_t * dma_streams,
             uint16_t nb_dma_streams )
```

References DMA_STREAM_FREE, DMA_STREAMS_MAX_LEN, ER_OUT_OF_RANGE, ER_SUCCESS, dma↵_stream_dev::state, and dma_scheduler_dev::streams.

Referenced by dma2_init_scheduler().

Here is the caller graph for this function:



4.14.4.6 dma_scheduler_release_stream()

```
util_error_t dma_scheduler_release_stream (
             dma_scheduler_dev_t * dma_scheduler,
             dma_stream_dev_t * dma_stream )
```

References DMA_STREAM_FREE, ENTER_CRITICAL, ER_SUCCESS, EXIT_CRITICAL, dma_scheduler_dev↩ ::free_stream_count, and dma_stream_dev::state.

4.14.4.7 dma_scheduler_request_stream()

```
dma_stream_dev_t* dma_scheduler_request_stream (
             dma_scheduler_dev_t * dma_scheduler )
```

References DMA_STREAM_BUSY, DMA_STREAM_FREE, ENTER_CRITICAL, EXIT_CRITICAL, dma_↩ scheduler_dev::free_stream_count, dma_stream_dev::state, dma_scheduler_dev::stream_count, and dma_↩ scheduler_dev::streams.

4.14.4.8 dma_start_stream()

```
util_error_t dma_start_stream (
             dma_stream_dev_t * stream,
             dma_stream_config_t config )
```

References dma_stream_config::direction, dma_stream_dev::dma, dma_stream_dev::dma_stream, dma_stream↩ _dev::dmamux_channel, dma_stream_config::dmamux_request_number, ER_OUT_OF_RANGE, dma_stream↩ _config::m0_addr, dma_stream_config::m1_addr, dma_stream_dev::number, dma_stream_config::p_addr, dma↩ _stream_config::peripheral_flow_control, dma_stream_config::priority, dma_stream_config::transfer_cplt, dma↩ _stream_dev::transfer_cplt, dma_stream_config::transfer_error, dma_stream_dev::transfer_error, dma_stream↩ _config::transfer_half, dma_stream_dev::transfer_half, dma_stream_config::transfer_size, dma_stream_config↩ ::user_context, dma_stream_dev::user_context, and WRITE_IN_REG.

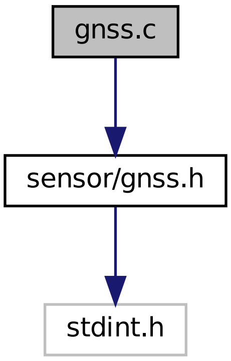
4.14.4.9 dma_stop_stream()

```
util_error_t dma_stop_stream (
             dma_stream_dev_t * stream )
```

4.15 gnss.c File Reference

```
#include <sensor/gnss.h>
```

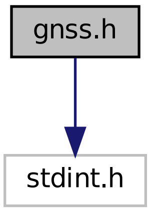
Include dependency graph for gnss.c:



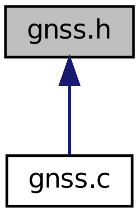
4.16 gnss.h File Reference

```
#include <stdint.h>
```

Include dependency graph for gnss.h:



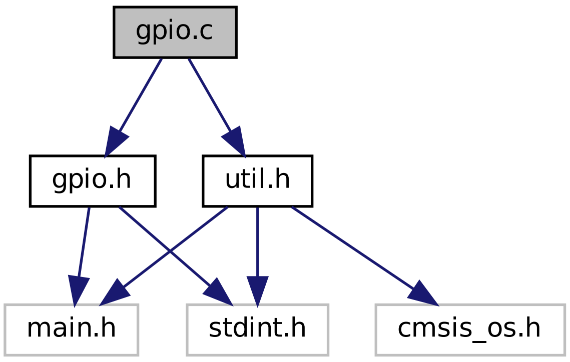
This graph shows which files directly or indirectly include this file:



4.17 gpio.c File Reference

```
#include "gpio.h"
#include <util.h>
```

Include dependency graph for gpio.c:



Functions

- uint8_t gpio_get (GPIO_TypeDef ∗gpio, uint16_t pin)
- void gpio_set (GPIO_TypeDef ∗gpio, uint16_t pin)
- void gpio_clr (GPIO_TypeDef ∗gpio, uint16_t pin)
- void gpio_cfg (GPIO_TypeDef ∗gpio, uint16_t pins, gpio_config_t cfg)

4.17.1 Function Documentation

4.17.1.1 gpio_cfg()

```
void gpio_cfg (
            GPIO_TypeDef * gpio,
            uint16_t pins,
            gpio_config_t cfg )
```

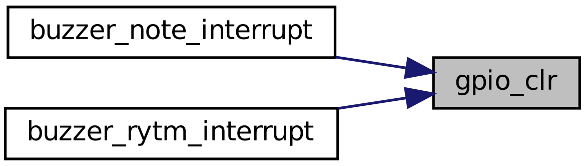
References gpio_config::alternate, gpio_config::drive, gpio_config::mode, gpio_config::speed, and WRITE_IN_↩ REG.

4.17.1.2 gpio_clr()

```
void gpio_clr (
            GPIO_TypeDef * gpio,
            uint16_t pin )
```

Referenced by buzzer_note_interrupt(), and buzzer_rytm_interrupt().

Here is the caller graph for this function:



4.17.1.3 gpio_get()

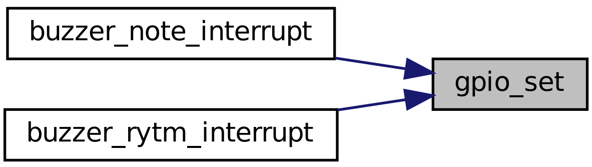
```
uint8_t gpio_get (
            GPIO_TypeDef * gpio,
            uint16_t pin )
```

4.17.1.4 gpio_set()

```
void gpio_set (
             GPIO_TypeDef * gpio,
             uint16_t pin )
```

Referenced by buzzer_note_interrupt(), and buzzer_rytm_interrupt().

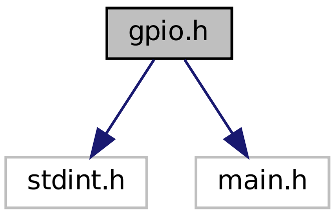
Here is the caller graph for this function:



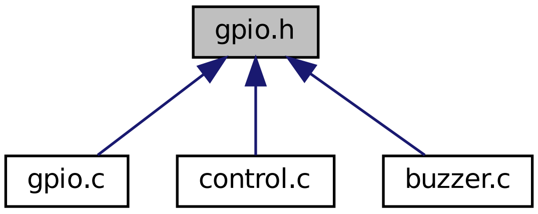
4.18 gpio.h File Reference

```
#include <stdint.h>
#include <main.h>
```

Include dependency graph for gpio.h:



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



Data Structures

- struct gpio_config

Typedefs

- typedef enum gpio_drive gpio_drive_t
- typedef enum gpio_mode gpio_mode_t
- typedef enum gpio_bias gpio_bias_t
- typedef struct gpio_config gpio_config_t

Enumerations

- enum gpio_drive { GPIO_DRIVE_PP = 0b0 , GPIO_DRIVE_OD = 0b1 }
- enum gpio_mode { GPIO_MODE_IN = 0b00 , GPIO_MODE_OUT = 0b01 , GPIO_MODE_ALT = 0b10 , GPIO_MODE_ANA = 0b11 }
- enum gpio_bias { GPIO_BIAS_NONE = 0b00 , GPIO_BIAS_HIGH = 0b01 , GPIO_BIAS_LOW = 0b10 }

Functions

- void gpio_set (GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin)
- void gpio_clr (GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin)
- uint8_t gpio_get (GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin)
- void gpio_cfg (GPIO_TypeDef ∗GPIOx, uint16_t GPIO_Pin, gpio_config_t cfg)

4.18.1 Typedef Documentation

4.18.1.1 gpio_bias_t

```
typedef enum gpio_bias gpio_bias_t
```

4.18.1.2 gpio_config_t

```
typedef struct gpio_config gpio_config_t
```

4.18.1.3 gpio_drive_t

```
typedef enum gpio_drive gpio_drive_t
```

4.18.1.4 gpio_mode_t

```
typedef enum gpio_mode gpio_mode_t
```

4.18.2 Enumeration Type Documentation

4.18.2.1 gpio_bias

```
enum gpio_bias
```

Enumerator

GPIO_BIAS_NONE	
GPIO_BIAS_HIGH	
GPIO_BIAS_LOW	

4.18.2.2 gpio_drive

```
enum gpio_drive
```

Enumerator

GPIO_DRIVE_PP	
GPIO_DRIVE_OD	

4.18.2.3 gpio_mode

```
enum gpio_mode
```

Enumerator

GPIO_MODE_IN	
GPIO_MODE_OUT	
GPIO_MODE_ALT	
GPIO_MODE_ANA	

4.18.3 Function Documentation

4.18.3.1 gpio_cfg()

```
void gpio_cfg (
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin,
            gpio_config_t cfg )
```

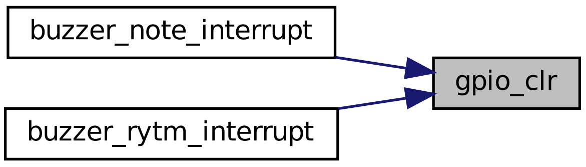
References gpio_config::alternate, gpio_config::drive, gpio_config::mode, gpio_config::speed, and WRITE_IN_↩
REG.

4.18.3.2 gpio_clr()

```
void gpio_clr (
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin )
```

Referenced by buzzer_note_interrupt(), and buzzer_rytm_interrupt().

Here is the caller graph for this function:



4.18.3.3 gpio_get()

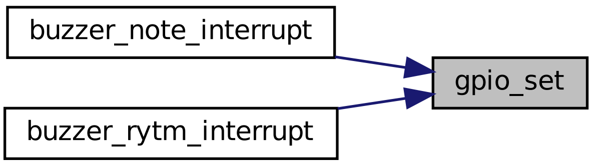
```
uint8_t gpio_get (
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin )
```

4.18.3.4 gpio_set()

```
void gpio_set (
            GPIO_TypeDef * GPIOx,
            uint16_t GPIO_Pin )
```

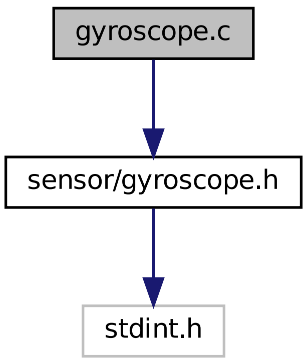
Referenced by buzzer_note_interrupt(), and buzzer_rytm_interrupt().

Here is the caller graph for this function:



4.19 gyroscope.c File Reference

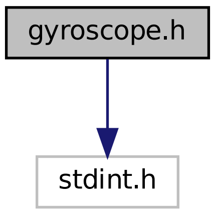
`#include <sensor/gyroscope.h>`
Include dependency graph for gyroscope.c:



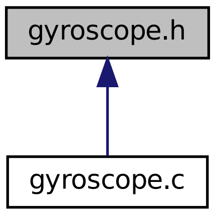
4.20 gyroscope.h File Reference

```
#include <stdint.h>
```

Include dependency graph for gyroscope.h:



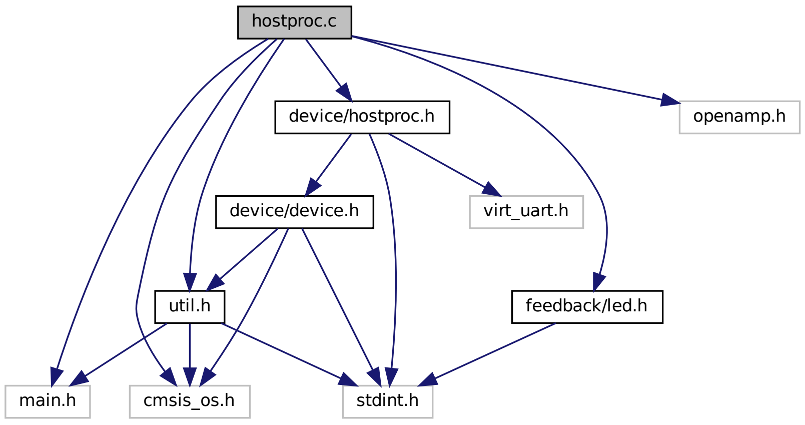
This graph shows which files directly or indirectly include this file:



4.21 hostproc.c File Reference

```
#include <main.h>
#include <cmsis_os.h>
#include <device/hostproc.h>
#include <feedback/led.h>
#include <openamp.h>
#include <util.h>
```

Include dependency graph for hostproc.c:



Data Structures

- struct hostproc_interface_context

Typedefs

- typedef struct hostproc_interface_context hostproc_interface_context_t

Functions

- void host_UART0_RX (VIRT_UART_HandleTypeDef ∗huart)
- util_error_t host_send (void ∗context, uint8_t ∗data, uint32_t len)
- util_error_t host_recv (void ∗context, uint8_t ∗data, uint32_t ∗len)
- device_interface_t ∗ hostproc_get_interface (void)
- device_t ∗ hostproc_get_device (void)
- util_error_t hostproc_init (void)

Variables

- static VIRT_UART_HandleTypeDef host_UART0
- static device_t hostproc_device
- static device_interface_t hostproc_interface
- static hostproc_interface_context_t hostproc_interface_context

4.21.1 Typedef Documentation

4.21.1.1 hostproc_interface_context_t

```
typedef struct hostproc_interface_context hostproc_interface_context_t
```

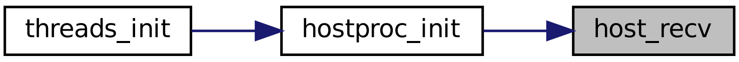
4.21.2 Function Documentation

4.21.2.1 host_recv()

```
util_error_t host_recv (
              void * context,
              uint8_t * data,
              uint32_t * len )
```

Referenced by hostproc_init().

Here is the caller graph for this function:



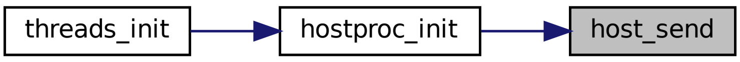
4.21.2.2 host_send()

```
util_error_t host_send (
              void * context,
              uint8_t * data,
              uint32_t len )
```

References ER_SUCCESS, hostproc_interface_context::rx_once, and hostproc_interface_context::uart.

Referenced by hostproc_init().

Here is the caller graph for this function:



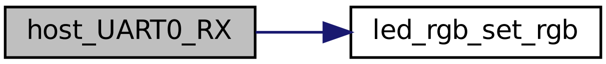
4.21.2.3 host_UART0_RX()

```
void host_UART0_RX (
            VIRT_UART_HandleTypeDef * huart )
```

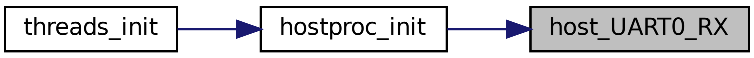
References led_rgb_set_rgb(), and hostproc_interface_context::rx_once.

Referenced by hostproc_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.2.4 hostproc_get_device()

```
device_t* hostproc_get_device (
            void )
```

References hostproc_device.

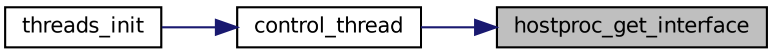
4.21.2.5 hostproc_get_interface()

```
device_interface_t* hostproc_get_interface (
            void )
```

References hostproc_interface.

Referenced by control_thread().

Here is the caller graph for this function:



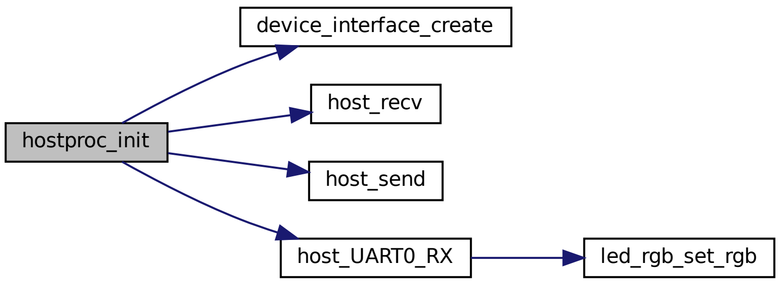
4.21.2.6 hostproc_init()

```
util_error_t hostproc_init (
            void )
```

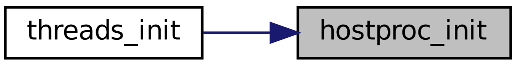
References device_interface_create(), ER_FAILURE, ER_SUCCESS, host_recv(), host_send(), host_UART0, host_UART0_RX(), hostproc_interface, hostproc_interface_context::rx_once, and hostproc_interface_context::uart.

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.21.3 Variable Documentation

4.21.3.1 host_UART0

```
VIRT_UART_HandleTypeDef host_UART0  [static]
```

Referenced by hostproc_init().

4.21.3.2 hostproc_device

```
device_t hostproc_device  [static]
```

Referenced by hostproc_get_device().

4.21.3.3 hostproc_interface

```
device_interface_t hostproc_interface  [static]
```

Referenced by control_thread(), hostproc_get_interface(), and hostproc_init().

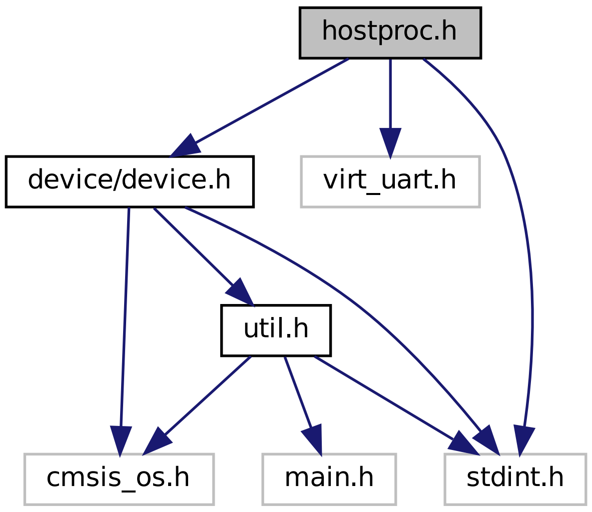
4.21.3.4 hostproc_interface_context

```
hostproc_interface_context_t hostproc_interface_context  [static]
```

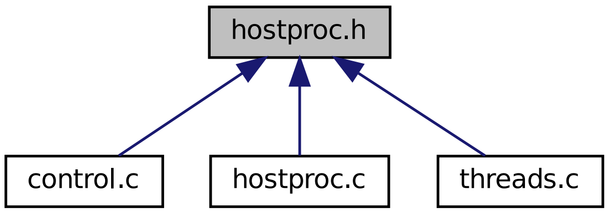
4.22 hostproc.h File Reference

```
#include <stdint.h>
#include <virt_uart.h>
#include <device/device.h>
```

Include dependency graph for hostproc.h:



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



Functions

- device_interface_t ∗ hostproc_get_interface (void)
- device_t ∗ hostproc_get_device (void)
- util_error_t hostproc_init (void)

4.22.1 Function Documentation

4.22.1.1 hostproc_get_device()

```
device_t* hostproc_get_device (
            void  )
```

References hostproc_device.

4.22.1.2 hostproc_get_interface()

```
device_interface_t* hostproc_get_interface (
            void  )
```

References hostproc_interface.

Referenced by control_thread().

Here is the caller graph for this function:

threads_init
control_thread
hostproc_get_interface

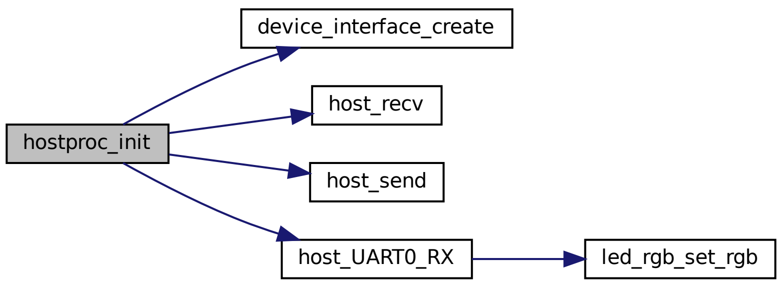
4.22.1.3 hostproc_init()

```
util_error_t hostproc_init (
            void  )
```

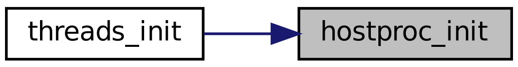
References device_interface_create(), ER_FAILURE, ER_SUCCESS, host_recv(), host_send(), host_UART0, host_UART0_RX(), hostproc_interface, hostproc_interface_context::rx_once, and hostproc_interface_context::uart.

Referenced by threads_init().

Here is the call graph for this function:



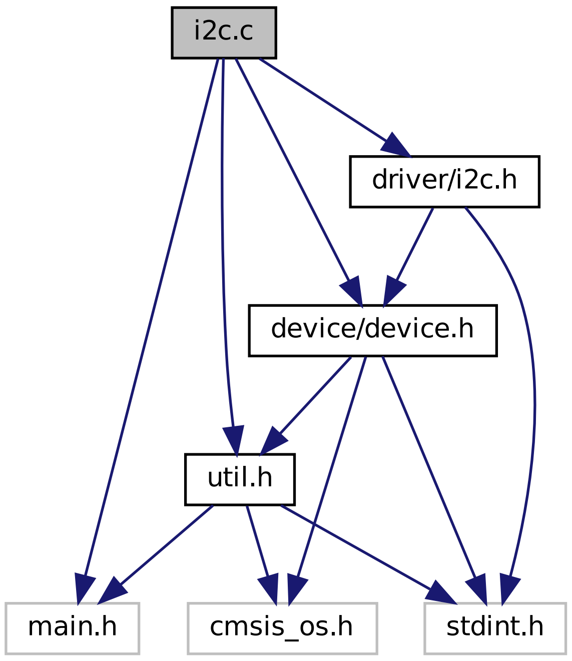
Here is the caller graph for this function:



4.23 i2c.c File Reference

```
#include <main.h>
#include <driver/i2c.h>
#include <device/device.h>
#include <util.h>
```

Include dependency graph for i2c.c:



Macros

- #define S1_I2C hi2c1
- #define S2_I2C hi2c2
- #define S3_I2C hi2c5

Functions

- void i2c_spi_guard (void)

  This funcion disables completely the SPI for hardware saveguard.
- device_interface_t ∗ i2c_get_sensor_interface (void)

  Getter for the interface associated with the sensors.
- void i2c_init (void)

  Initializer for the i2c subsystem. @deails This function initializes the three i2c interfaces which are present on the hostboards.

Variables

- device_interface_t sensor_interface

  This is the I2C Interface associated with the sensors.
- i2c_interface_context_t sensor_interface_context

4.23.1 Macro Definition Documentation

4.23.1.1 S1_I2C

```
#define S1_I2C hi2c1
```

4.23.1.2 S2_I2C

```
#define S2_I2C hi2c2
```

4.23.1.3 S3_I2C

```
#define S3_I2C hi2c5
```

4.23.2 Function Documentation

4.23.2.1 i2c_get_sensor_interface()

```
device_interface_t* i2c_get_sensor_interface (
             void  )
```

Getter for the interface associated with the sensors.

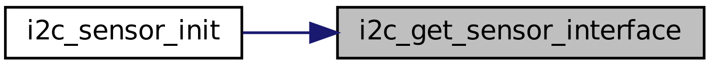
Returns

The pointer to the sensor interface object

References sensor_interface.

Referenced by i2c_sensor_init().

Here is the caller graph for this function:



4.23.2.2 i2c_init()

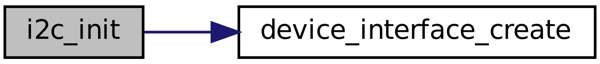
```
void i2c_init (
             void  )
```

Initializer for the i2c subsystem. @deails This function initializes the three i2c interfaces which are present on the hostboards.

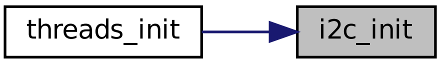
References device_interface_create(), ER_SUCCESS, sensor_interface, and sensor_interface_context.

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



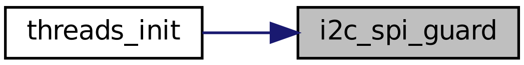
4.23.2.3 i2c_spi_guard()

```
void i2c_spi_guard (
             void  )
```

This funcion disables completely the SPI for hardware saveguard.

Referenced by threads_init().

Here is the caller graph for this function:



4.23.3 Variable Documentation

4.23.3.1 sensor_interface

```
device_interface_t sensor_interface
```

This is the I2C Interface associated with the sensors.

Referenced by i2c_get_sensor_interface(), and i2c_init().

4.23.3.2 sensor_interface_context

```
i2c_interface_context_t sensor_interface_context
```

Initial value:

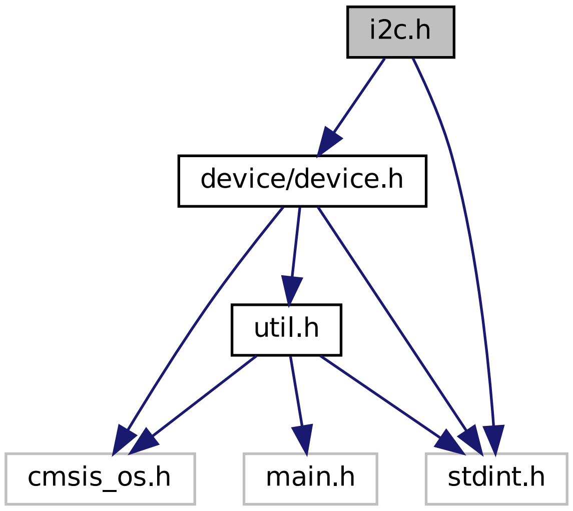
```
= {
        .i2c = &S2_I2C
}
```

Referenced by i2c_init().

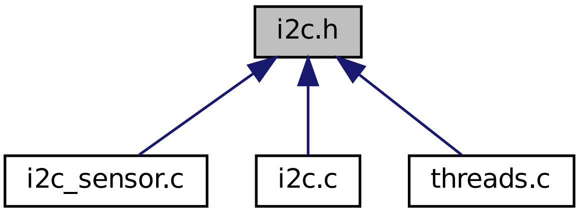
4.24 i2c.h File Reference

```
#include <stdint.h>
#include <device/device.h>
```

Include dependency graph for i2c.h:



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



Data Structures

- struct i2c_interface_context

  I2C interface context structure.

Typedefs

- typedef struct i2c_interface_context i2c_interface_context_t

  I2C interface context structure.

Functions

- void i2c_init (void)

  Initializer for the i2c subsystem. @deails This function initializes the three i2c interfaces which are present on the hostboards.
- void i2c_spi_guard (void)

  This funcion disables completely the SPI for hardware saveguard.
- device_interface_t ∗ i2c_get_sensor_interface (void)

  Getter for the interface associated with the sensors.

4.24.1 Typedef Documentation

4.24.1.1 i2c_interface_context_t

```
typedef struct i2c_interface_context i2c_interface_context_t
```

I2C interface context structure.

this only contains the HAL I2C handle pointer.

4.24.2 Function Documentation

4.24.2.1 i2c_get_sensor_interface()

```
device_interface_t* i2c_get_sensor_interface (
             void  )
```

Getter for the interface associated with the sensors.

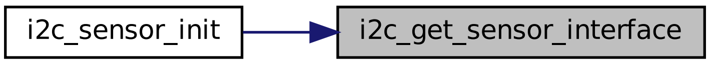
Returns

The pointer to the sensor interface object

References sensor_interface.

Referenced by i2c_sensor_init().

Here is the caller graph for this function:



4.24.2.2 i2c_init()

```
void i2c_init (
             void  )
```

Initializer for the i2c subsystem. @deails This function initializes the three i2c interfaces which are present on the hostboards.

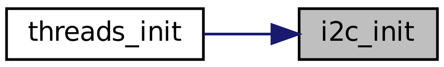
References device_interface_create(), ER_SUCCESS, sensor_interface, and sensor_interface_context.

Referenced by threads_init().

Here is the call graph for this function:

i2c_init
device_interface_create

Here is the caller graph for this function:



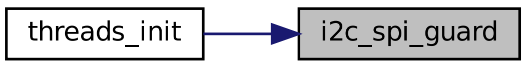
4.24.2.3 i2c_spi_guard()

```
void i2c_spi_guard (
             void  )
```

This funcion disables completely the SPI for hardware saveguard.

Referenced by threads_init().

Here is the caller graph for this function:

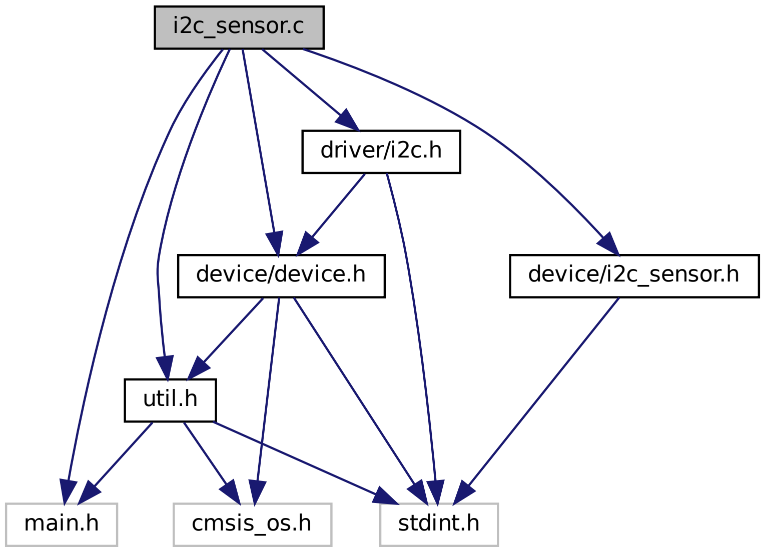


4.25 i2c_sensor.c File Reference

```
#include <main.h>
#include <device/device.h>
#include <device/i2c_sensor.h>
#include <driver/i2c.h>
```

```
#include <util.h>
```

Include dependency graph for i2c_sensor.c:



Data Structures

- struct i2c_sensor_context

Typedefs

- typedef struct i2c_sensor_context i2c_sensor_context_t

Functions

- util_error_t read_reg (void ∗context, device_interface_t ∗dev, uint32_t address, uint8_t ∗data, uint32_t data↩ _len)
- util_error_t write_reg (void ∗context, device_interface_t ∗dev, uint32_t address, uint8_t ∗data, uint32_t data↩ _len)
- device_t ∗ i2c_get_accelerometer (void)
- util_error_t i2c_sensor_init (void)

Variables

- static device_t i2c_accelerometer_device
- static device_t i2c_gyroscope_device
- static device_t i2c_barometer_device
- static i2c_sensor_context_t i2c_accelerometer_device_context
- static i2c_sensor_context_t i2c_gyroscope_device_context
- static i2c_sensor_context_t i2c_barometer_device_context

4.25.1 Typedef Documentation

4.25.1.1 i2c_sensor_context_t

```
typedef struct i2c_sensor_context i2c_sensor_context_t
```

4.25.2 Function Documentation

4.25.2.1 i2c_get_accelerometer()

```
device_t* i2c_get_accelerometer (
             void   )
```

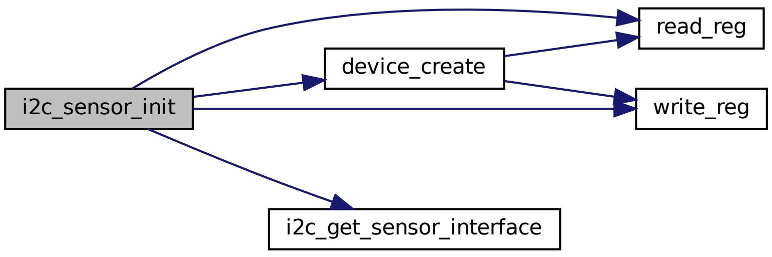
References i2c_accelerometer_device.

4.25.2.2 i2c_sensor_init()

```
util_error_t i2c_sensor_init (
             void   )
```

References device_create(), ER_SUCCESS, i2c_accelerometer_device, i2c_accelerometer_device_context, i2c←_get_sensor_interface(), read_reg(), and write_reg().

Here is the call graph for this function:



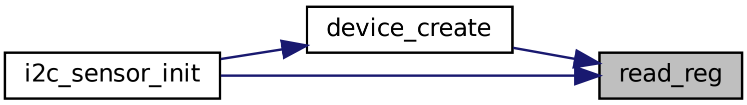
4.25.2.3 read_reg()

```
util_error_t read_reg (
             void * context,
             device_interface_t * dev,
             uint32_t address,
             uint8_t * data,
             uint32_t data_len )
```

References device_interface::context, ER_SUCCESS, and i2c_interface_context::i2c.

Referenced by device_create(), and i2c_sensor_init().

Here is the caller graph for this function:



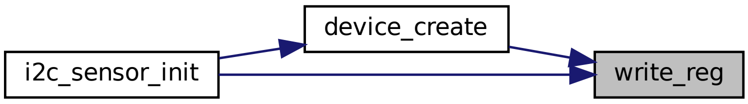
4.25.2.4 write_reg()

```
util_error_t write_reg (
             void * context,
             device_interface_t * dev,
             uint32_t address,
             uint8_t * data,
             uint32_t data_len )
```

References device_interface::context, ER_SUCCESS, and i2c_interface_context::i2c.

Referenced by device_create(), and i2c_sensor_init().

Here is the caller graph for this function:



4.25.3 Variable Documentation

4.25.3.1 i2c_accelerometer_device

```
device_t i2c_accelerometer_device  [static]
```

Referenced by i2c_get_accelerometer(), and i2c_sensor_init().

4.25.3.2 i2c_accelerometer_device_context

```
i2c_sensor_context_t i2c_accelerometer_device_context  [static]
```

Initial value:

```
= {
        .device_address = 0x68
}
```

Referenced by i2c_sensor_init().

4.25.3.3 i2c_barometer_device

```
device_t i2c_barometer_device  [static]
```

4.25.3.4 i2c_barometer_device_context

```
i2c_sensor_context_t i2c_barometer_device_context  [static]
```

Initial value:

```
= {
        .device_address = 0x18
}
```

4.25.3.5 i2c_gyroscope_device

```
device_t i2c_gyroscope_device  [static]
```

4.25.3.6 i2c_gyroscope_device_context

```
i2c_sensor_context_t i2c_gyroscope_device_context  [static]
```

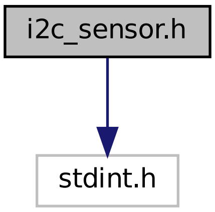
Initial value:

```
= {
        .device_address = 0x68
}
```

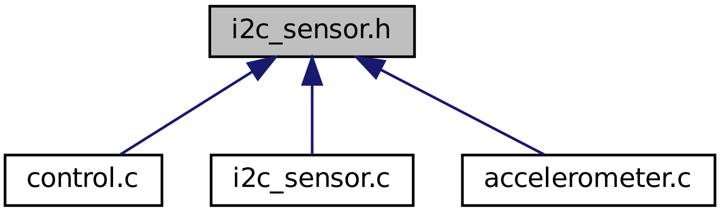
4.26 i2c_sensor.h File Reference

```
#include <stdint.h>
```

Include dependency graph for i2c_sensor.h:



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



Functions

- util_error_t i2c_sensor_init (void)

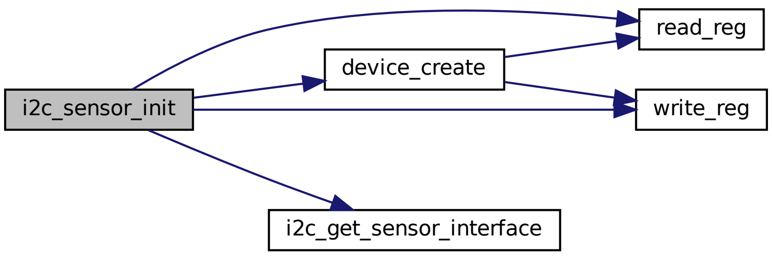
4.26.1 Function Documentation

4.26.1.1 i2c_sensor_init()

```
util_error_t i2c_sensor_init (
             void  )
```

References device_create(), ER_SUCCESS, i2c_accelerometer_device, i2c_accelerometer_device_context, i2c↩
_get_sensor_interface(), read_reg(), and write_reg().

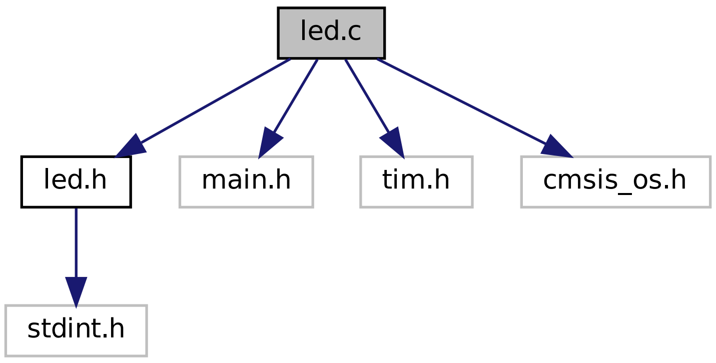
Here is the call graph for this function:



4.27 led.c File Reference

```
#include "led.h"
#include <main.h>
#include <tim.h>
#include <cmsis_os.h>
```

Include dependency graph for led.c:



Macros

- #define LED_TIM htim3
- #define LED_MAX (0xff)

Typedefs

- typedef enum led_blick_state led_blink_state_t

Enumerations

- enum led_blick_state { LED_ON , LED_FAINT , LED_OFF }

Functions

- void led_feedback_init (void)

  Initialize the feedback module.
- void led_rgb_init (void)

  Initialize the RGB LED.
- void led_rgb_set_rgb (uint8_t r, uint8_t g, uint8_t b)

  Set RBG LED color using r, g, b values.
- void led_rgb_set_color (led_color_t color)

  Set RBG LED color using color structure.
- void led_rgb_thread (__attribute__((unused)) void ∗arg)

Variables

- static led_blink_state_t blink_sequence [ ]
- static const int blink_sequence_len = sizeof(blink_sequence)/sizeof(led_blink_state_t)
- static led_color_t color_sequence [ ]
- static const int color_sequence_len = sizeof(color_sequence)/sizeof(led_color_t)

4.27.1 Macro Definition Documentation

4.27.1.1 LED_MAX

```
#define LED_MAX (0xff)
```

4.27.1.2 LED_TIM

```
#define LED_TIM htim3
```

4.27.2 Typedef Documentation

4.27.2.1 led_blink_state_t

```
typedef enum led_blick_state led_blink_state_t
```

4.27.3 Enumeration Type Documentation

4.27.3.1 led_blick_state

```
enum led_blick_state
```

Enumerator

LED_ON	
LED_FAINT	
LED_OFF	

4.27.4 Function Documentation

4.27.4.1 led_feedback_init()

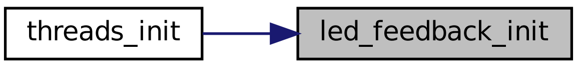
```
void led_feedback_init (
            void )
```

Initialize the feedback module.

This will initialize a board to accept a feedback board on the S3 socket.

Referenced by threads_init().

Here is the caller graph for this function:



4.27.4.2 led_rgb_init()

```
void led_rgb_init (
             void  )
```

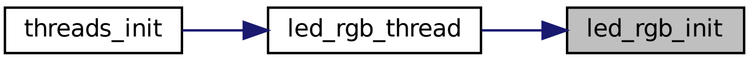
Initialize the RGB LED.

Starts the PWM channels connected to the RGB led for user feedback.

References LED_MAX, and LED_TIM.

Referenced by led_rgb_thread().

Here is the caller graph for this function:



4.27.4.3 led_rgb_set_color()

```
void led_rgb_set_color (
             led_color_t color )
```

Set RBG LED color using color structure.

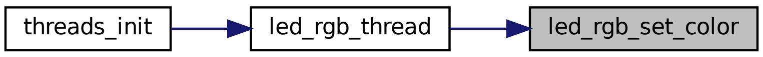
Parameters

color	Color structure, defines the color to be set.

References led_color::b, led_color::g, LED_TIM, and led_color::r.

Referenced by led_rgb_thread().

Here is the caller graph for this function:



4.27.4.4 led_rgb_set_rgb()

```
void led_rgb_set_rgb (
             uint8_t r,
             uint8_t g,
             uint8_t b )
```

Set RBG LED color using r, g, b values.

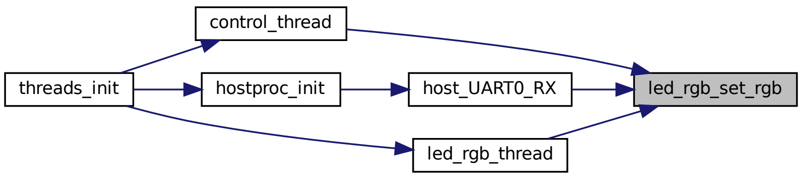
Parameters

r	Red channel value.
g	Green channel value.
b	Blue channel value.

References LED_TIM.

Referenced by control_thread(), host_UART0_RX(), and led_rgb_thread().

Here is the caller graph for this function:



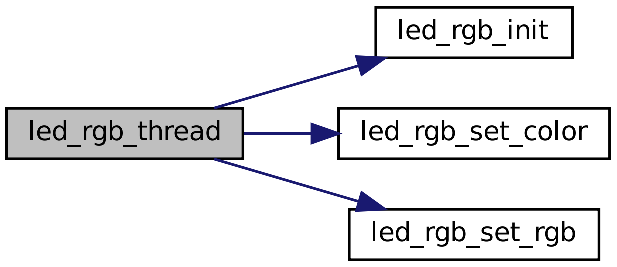
4.27.4.5 led_rgb_thread()

```
void led_rgb_thread (
             __attribute__((unused)) void * arg )
```

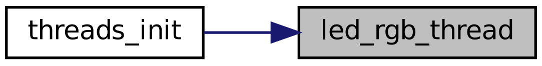
References blink_sequence, blink_sequence_len, color_sequence, color_sequence_len, LED_BLACK, led_blue, LED_FAINT, LED_OFF, LED_ON, led_rgb_init(), led_rgb_set_color(), and led_rgb_set_rgb().

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.5 Variable Documentation

4.27.5.1 blink_sequence

```
led_blink_state_t blink_sequence[ ]  [static]
```

Initial value:

```
= {
        LED_ON,
        LED_FAINT,
        LED_ON,
        LED_FAINT,
        LED_ON,
        LED_OFF
}
```

Referenced by led_rgb_thread().

4.27.5.2 blink_sequence_len

```
const int blink_sequence_len = sizeof(blink_sequence)/sizeof(led_blink_state_t)  [static]
```

Referenced by led_rgb_thread().

4.27.5.3 color_sequence

```
led_color_t color_sequence[]  [static]
```

Initial value:

```
= {
        led_green,
        led_red,
        led_blue,
        led_red,
}
```

Referenced by led_rgb_thread().

4.27.5.4 color_sequence_len

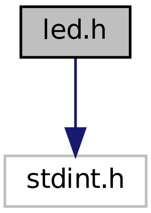
```
const int color_sequence_len = sizeof(color_sequence)/sizeof(led_color_t)  [static]
```

Referenced by led_rgb_thread().

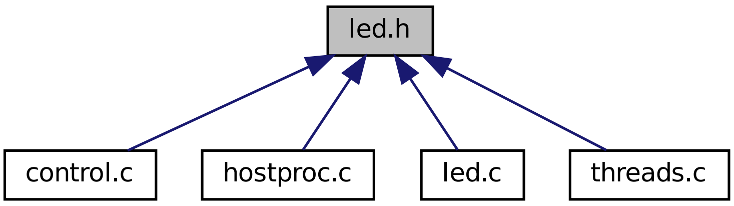
4.28 led.h File Reference

```
#include <stdint.h>
```

Include dependency graph for led.h:



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



Data Structures

- struct led_color

Macros

- #define LED_RED 0xff, 0x00, 0x00
- #define LED_GREEN 0x00, 0xff, 0x00
- #define LED_BLUE 0x00, 0x00, 0xff
- #define LED_ORANGE 0x7f, 0x0f, 0x07
- #define LED_YELLOW 0xff, 0x1f, 0x07
- #define LED_TEAL 0x00, 0x7f, 0x7f
- #define LED_PINK 0x7f, 0x00, 0x7f
- #define LED_LILA 0xff, 0x03, 0x4f
- #define LED_BLACK 0x00, 0x00, 0x00
- #define LED_WHITE 0xff, 0xff, 0xff

Typedefs

- typedef struct led_color led_color_t

Functions

- void led_rgb_init (void)

  Initialize the RGB LED.
- void led_rgb_set_color (led_color_t color)

  Set RBG LED color using color structure.
- void led_rgb_set_rgb (uint8_t r, uint8_t g, uint8_t b)

  Set RBG LED color using r, g, b values.
- void led_feedback_init (void)

  Initialize the feedback module.
- void led_rgb_thread (void ∗arg)

Variables

- static const led_color_t led_red
- static const led_color_t led_green
- static const led_color_t led_blue
- static const led_color_t led_black

4.28.1 Macro Definition Documentation

4.28.1.1 LED_BLACK

```
#define LED_BLACK 0x00, 0x00, 0x00
```

4.28.1.2 LED_BLUE

```
#define LED_BLUE 0x00, 0x00, 0xff
```

4.28.1.3 LED_GREEN

```
#define LED_GREEN 0x00, 0xff, 0x00
```

4.28.1.4 LED_LILA

```
#define LED_LILA 0xff, 0x03, 0x4f
```

4.28.1.5 LED_ORANGE

```
#define LED_ORANGE 0x7f, 0x0f, 0x07
```

4.28.1.6 LED_PINK

```
#define LED_PINK 0x7f, 0x00, 0x7f
```

4.28.1.7 LED_RED

```
#define LED_RED 0xff, 0x00, 0x00
```

4.28.1.8 LED_TEAL

```
#define LED_TEAL 0x00, 0x7f, 0x7f
```

4.28.1.9 LED_WHITE

```
#define LED_WHITE 0xff, 0xff, 0xff
```

4.28.1.10 LED_YELLOW

```
#define LED_YELLOW 0xff, 0x1f, 0x07
```

4.28.2 Typedef Documentation

4.28.2.1 led_color_t

```
typedef struct led_color led_color_t
```

4.28.3 Function Documentation

4.28.3.1 led_feedback_init()

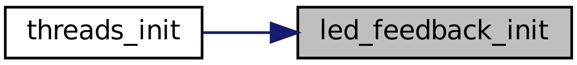
```
void led_feedback_init (
             void  )
```

Initialize the feedback module.

This will initialize a board to accept a feedback board on the S3 socket.

Referenced by threads_init().

Here is the caller graph for this function:



4.28.3.2 led_rgb_init()

```
void led_rgb_init (
             void  )
```

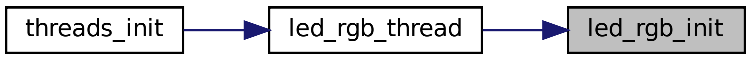
Initialize the RGB LED.

Starts the PWM channels connected to the RGB led for user feedback.

References LED_MAX, and LED_TIM.

Referenced by led_rgb_thread().

Here is the caller graph for this function:



4.28.3.3 led_rgb_set_color()

```
void led_rgb_set_color (
             led_color_t color )
```

Set RBG LED color using color structure.

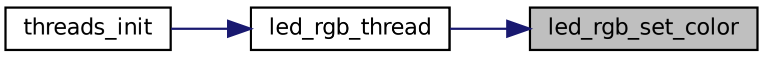
Parameters

color	Color structure, defines the color to be set.

References led_color::b, led_color::g, LED_TIM, and led_color::r.

Referenced by led_rgb_thread().

Here is the caller graph for this function:



4.28.3.4 led_rgb_set_rgb()

```
void led_rgb_set_rgb (
            uint8_t r,
            uint8_t g,
            uint8_t b )
```

Set RBG LED color using r, g, b values.

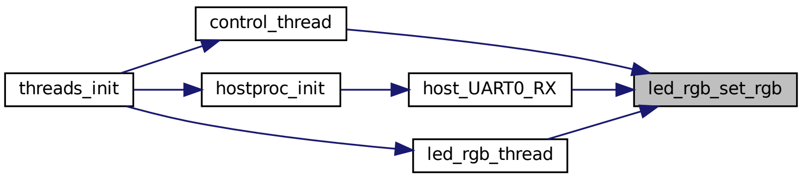
Parameters

r	Red channel value.
g	Green channel value.
b	Blue channel value.

References LED_TIM.

Referenced by control_thread(), host_UART0_RX(), and led_rgb_thread().

Here is the caller graph for this function:



4.28.3.5 led_rgb_thread()

```
void led_rgb_thread (
             void * arg )
```

4.28.4 Variable Documentation

4.28.4.1 led_black

```
const led_color_t led_black  [static]
```

Initial value:

```
= {
        .r = 0x00,
        .g = 0x00,
        .b = 0x00
}
```

4.28.4.2 led_blue

```
const led_color_t led_blue  [static]
```

Initial value:

```
= {
        .r = 0x00,
        .g = 0x00,
        .b = 0xff
}
```

Referenced by led_rgb_thread().

4.28.4.3 led_green

```
const led_color_t led_green  [static]
```

Initial value:

```
= {
        .r = 0x00,
        .g = 0xff,
        .b = 0x00
}
```

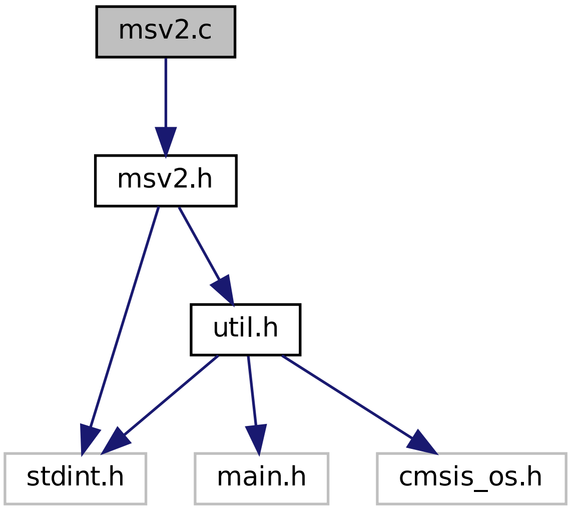
4.28.4.4 led_red

```
const led_color_t led_red [static]
```

Initial value:
```
= {
        .r = 0xff,
        .g = 0x00,
        .b = 0x00
}
```

4.29 msv2.c File Reference

```
#include "msv2.h"
```
Include dependency graph for msv2.c:



Macros

- #define DLE (0x90)
- #define STX (0x02)

Functions

- static uint16_t calc_field_CRC (uint16_t ∗p_data_array, uint16_t length)

    Compute the CRC for a data packet.
- void msv2_init (MSV2_INST_t ∗msv2)

    Initialize a msv2 packet creator instance.
- uint16_t msv2_create_frame (MSV2_INST_t ∗msv2, uint8_t opcode, uint8_t data_len, uint8_t ∗data)

    Generate an msv2 packet.

- MSV2_ERROR_t msv2_decode_fragment (MSV2_INST_t ∗msv2, uint8_t d)

    Decode an msv2 fragment (one byte at a time)
- uint8_t ∗ msv2_rx_data (MSV2_INST_t ∗msv2)

    Getter for the pointer to the rx data array contained in the msv2 packet creator.
- uint8_t ∗ msv2_tx_data (MSV2_INST_t ∗msv2)

    Getter for the pointer to the tx data array contained in the msv2 packet creator.

4.29.1 Macro Definition Documentation

4.29.1.1 DLE

```
#define DLE (0x90)
```

4.29.1.2 STX

```
#define STX (0x02)
```

4.29.2 Function Documentation

4.29.2.1 calc_field_CRC()

```
uint16_t calc_field_CRC (
            uint16_t * p_data_array,
            uint16_t length )  [static]
```

Compute the CRC for a data packet.

Function taken from Maxon Serial V2 specification manual.

Parameters

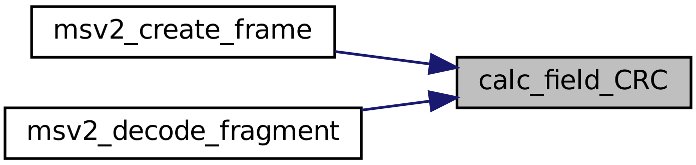
p_data_array	Pointer to the data array of the packet.
length	Length of the packet

Note

The p_data_array data must contain two empty bytes [0x00, 0x00] at the end, where the CRC goes.

Referenced by msv2_create_frame(), and msv2_decode_fragment().

Here is the caller graph for this function:



4.29.2.2 msv2_create_frame()

```
uint16_t msv2_create_frame (
             MSV2_INST_t * msv2,
             uint8_t opcode,
             uint8_t data_len,
             uint8_t * data )
```

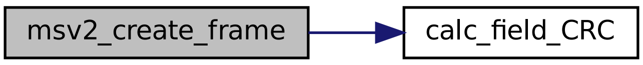
Generate an msv2 packet.

Parameters

msv2	Pointer to the msv2 packet creator instance.
opcode	Opcode to be placed in the packet header.
data_len	Length of the packet data payload.
data	Data to be placed in the packet payload.

References calc_field_CRC(), MSV2_TX_DATA::crc_data, MSV2_TX_DATA::data, MSV2_TX_DATA::data_len, DLE, MSV2_TX_DATA::opcode, STX, and MSV2_INST::tx.

Here is the call graph for this function:



4.29.2.3 msv2_decode_fragment()

```
MSV2_ERROR_t msv2_decode_fragment (
            MSV2_INST_t * msv2,
            uint8_t d )
```

Decode an msv2 fragment (one byte at a time)

Parameters

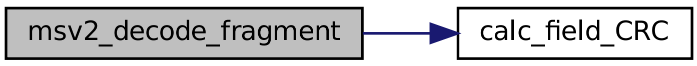
msv2	Pointer to the msv2 packet creator instance.
d	Fragment to be decoded.

Note

This function is to be called upon reception of a byte.

References calc_field_CRC(), MSV2_RX_DATA::counter, MSV2_RX_DATA::crc, MSV2_RX_DATA::crc_data, MSV2_RX_DATA::data, MSV2_RX_DATA::data_len, DLE, MSV2_RX_DATA::escape, MSV2_RX_DATA::length, MSV2_PROGRESS, MSV2_SUCCESS, MSV2_WRONG_CRC, MSV2_RX_DATA::opcode, MSV2_INST::rx, MSV2_RX_DATA::state, STX, WAITING_CRC1, WAITING_CRC2, WAITING_DATA, WAITING_DLE, WAITING_↩
LEN, WAITING_OPCODE, and WAITING_STX.

Here is the call graph for this function:



4.29.2.4 msv2_init()

```
void msv2_init (
            MSV2_INST_t * msv2 )
```

Initialize a msv2 packet creator instance.

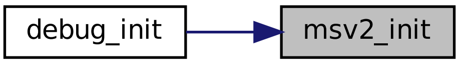
Parameters

msv2	Pointer to the msv2 instance to be initialized.

References MSV2_INST::id.

Referenced by debug_init().

Here is the caller graph for this function:



4.29.2.5 msv2_rx_data()

```
uint8_t* msv2_rx_data (
            MSV2_INST_t * msv2 )
```

Getter for the pointer to the rx data array contained in the msv2 packet creator.

Parameters

msv2	Pointer to the msv2 packet creator instance.

References MSV2_RX_DATA::data, and MSV2_INST::rx.

4.29.2.6 msv2_tx_data()

```
uint8_t* msv2_tx_data (
            MSV2_INST_t * msv2 )
```

Getter for the pointer to the tx data array contained in the msv2 packet creator.

Parameters

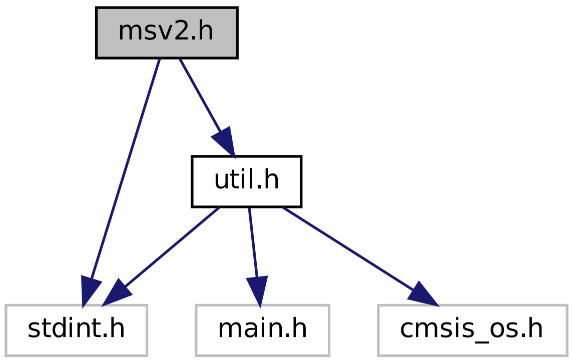
msv2	Pointer to the msv2 packet creator instance.

References MSV2_TX_DATA::data, and MSV2_INST::tx.

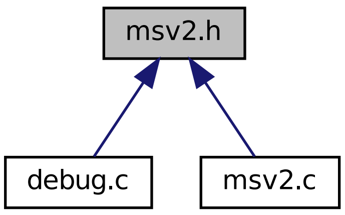
4.30 msv2.h File Reference

```
#include <util.h>
#include <stdint.h>
```

Include dependency graph for msv2.h:



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



Data Structures

- struct MSV2_RX_DATA
- struct MSV2_TX_DATA
- struct MSV2_INST

Macros

- #define MSV2_MAX_FRAME_LEN (1024)
- #define MSV2_MAX_DATA_LEN (512)

Typedefs

- typedef enum MSV2_ERROR MSV2_ERROR_t

  Msv2 packet creator return codes.
- typedef enum MSV2_DECODE_STATE MSV2_DECODE_STATE_t

  Msv2 packet decoded state.
- typedef struct MSV2_RX_DATA MSV2_RX_DATA_t
- typedef struct MSV2_TX_DATA MSV2_TX_DATA_t
- typedef struct MSV2_INST MSV2_INST_t

Enumerations

- enum MSV2_ERROR { MSV2_SUCCESS = 0 , MSV2_PROGRESS , MSV2_WRONG_CRC , MSV2_ERROR }

  Msv2 packet creator return codes.
- enum MSV2_DECODE_STATE {
  WAITING_DLE , WAITING_STX , WAITING_OPCODE , WAITING_LEN ,
  WAITING_DATA , WAITING_CRC1 , WAITING_CRC2 }

  Msv2 packet decoded state.

Functions

- MSV2_ERROR_t msv2_decode_fragment (MSV2_INST_t ∗msv2, uint8_t d)

  Decode an msv2 fragment (one byte at a time)
- void msv2_init (MSV2_INST_t ∗msv2)

  Initialize a msv2 packet creator instance.
- uint16_t msv2_create_frame (MSV2_INST_t ∗msv2, uint8_t opcode, uint8_t data_len, uint8_t ∗data)

  Generate an msv2 packet.
- uint8_t ∗ msv2_rx_data (MSV2_INST_t ∗msv2)

  Getter for the pointer to the rx data array contained in the msv2 packet creator.
- uint8_t ∗ msv2_tx_data (MSV2_INST_t ∗msv2)

  Getter for the pointer to the tx data array contained in the msv2 packet creator.

4.30.1 Macro Definition Documentation

4.30.1.1 MSV2_MAX_DATA_LEN

```
#define MSV2_MAX_DATA_LEN (512)
```

4.30.1.2 MSV2_MAX_FRAME_LEN

```
#define MSV2_MAX_FRAME_LEN (1024)
```

4.30.2 Typedef Documentation

4.30.2.1 MSV2_DECODE_STATE_t

`typedef enum MSV2_DECODE_STATE MSV2_DECODE_STATE_t`

Msv2 packet decoded state.

4.30.2.2 MSV2_ERROR_t

`typedef enum MSV2_ERROR MSV2_ERROR_t`

Msv2 packet creator return codes.

4.30.2.3 MSV2_INST_t

`typedef struct MSV2_INST MSV2_INST_t`

4.30.2.4 MSV2_RX_DATA_t

`typedef struct MSV2_RX_DATA MSV2_RX_DATA_t`

4.30.2.5 MSV2_TX_DATA_t

`typedef struct MSV2_TX_DATA MSV2_TX_DATA_t`

4.30.3 Enumeration Type Documentation

4.30.3.1 MSV2_DECODE_STATE

`enum MSV2_DECODE_STATE`

Msv2 packet decoded state.

Enumerator

WAITING_DLE	Waiting for data Link Escape
WAITING_STX	Waiting for Start of Text
WAITING_OPCODE	Waiting for Opcode
WAITING_LEN	Waiting for Packet Length
WAITING_DATA	Waiting for Data
WAITING_CRC1	Waiting for CRC Byte 1
WAITING_CRC2	Waiting for CRC Byte 2

4.30.3.2 MSV2_ERROR

```
enum MSV2_ERROR
```

Msv2 packet creator return codes.

Enumerator

MSV2_SUCCESS	Packet sucessfully decoded/encoded
MSV2_PROGRESS	Packed decoding in progress
MSV2_WRONG_CRC	Packet decoding error due to wrong crc
MSV2_ERROR	generic error

4.30.4 Function Documentation

4.30.4.1 msv2_create_frame()

```
uint16_t msv2_create_frame (
            MSV2_INST_t * msv2,
            uint8_t opcode,
            uint8_t data_len,
            uint8_t * data )
```

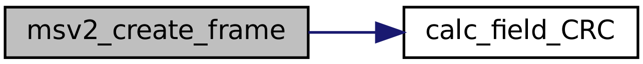
Generate an msv2 packet.

Parameters

msv2	Pointer to the msv2 packet creator instance.
opcode	Opcode to be placed in the packet header.
data_len	Length of the packet data payload.
data	Data to be placed in the packet payload.

References calc_field_CRC(), MSV2_TX_DATA::crc_data, MSV2_TX_DATA::data, MSV2_TX_DATA::data_len, DLE, MSV2_TX_DATA::opcode, STX, and MSV2_INST::tx.

Here is the call graph for this function:



4.30.4.2 msv2_decode_fragment()

```
MSV2_ERROR_t msv2_decode_fragment (
             MSV2_INST_t * msv2,
             uint8_t d )
```

Decode an msv2 fragment (one byte at a time)

Parameters

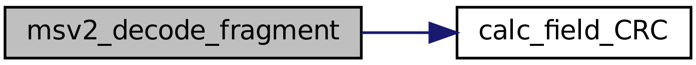
msv2	Pointer to the msv2 packet creator instance.
d	Fragment to be decoded.

Note

This function is to be called upon reception of a byte.

References calc_field_CRC(), MSV2_RX_DATA::counter, MSV2_RX_DATA::crc, MSV2_RX_DATA::crc_data, MSV2_RX_DATA::data, MSV2_RX_DATA::data_len, DLE, MSV2_RX_DATA::escape, MSV2_RX_DATA::length, MSV2_PROGRESS, MSV2_SUCCESS, MSV2_WRONG_CRC, MSV2_RX_DATA::opcode, MSV2_INST::rx, MSV2_RX_DATA::state, STX, WAITING_CRC1, WAITING_CRC2, WAITING_DATA, WAITING_DLE, WAITING_↵ LEN, WAITING_OPCODE, and WAITING_STX.

Here is the call graph for this function:



4.30.4.3 msv2_init()

```
void msv2_init (
            MSV2_INST_t * msv2 )
```

Initialize a msv2 packet creator instance.

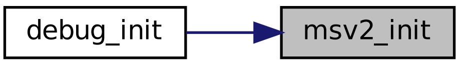
Parameters

msv2	Pointer to the msv2 instance to be initialized.

References MSV2_INST::id.

Referenced by debug_init().

Here is the caller graph for this function:



4.30.4.4 msv2_rx_data()

```
uint8_t* msv2_rx_data (
            MSV2_INST_t * msv2 )
```

Getter for the pointer to the rx data array contained in the msv2 packet creator.

Parameters

msv2	Pointer to the msv2 packet creator instance.

References MSV2_RX_DATA::data, and MSV2_INST::rx.

4.30.4.5 msv2_tx_data()

```
uint8_t* msv2_tx_data (
            MSV2_INST_t * msv2 )
```

Getter for the pointer to the tx data array contained in the msv2 packet creator.

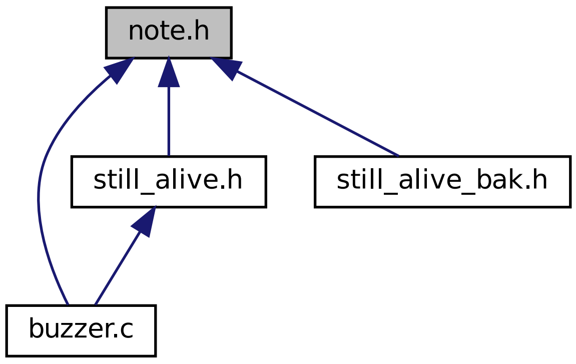
Parameters

msv2	Pointer to the msv2 packet creator instance.

References MSV2_TX_DATA::data, and MSV2_INST::tx.

4.31 note.h File Reference

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



Data Structures

- struct note

Macros

- #define T1_4 1
- #define T1_2 2
- #define T1 4
- #define T1_1_2 6
- #define T2 8
- #define T4 16
- #define C0 163
- #define C0H 173
- #define D0 183
- #define D0H 194
- #define E0 206
- #define F0 218
- #define F0H 231
- #define G0 245
- #define G0H 259

- #define A0 275
- #define A0H 291
- #define B0 308
- #define C1 327
- #define C1H 346
- #define D1 367
- #define D1H 388
- #define E1 412
- #define F1 436
- #define F1H 462
- #define G1 490
- #define G1H 519
- #define A1 550
- #define A1H 582
- #define B1 617
- #define C2 654
- #define C2H 693
- #define D2 734
- #define D2H 777
- #define E2 824
- #define F2 873
- #define F2H 925
- #define G2 980
- #define G2H 1038
- #define A2 1100
- #define A2H 1165
- #define B2 1234
- #define C3 1308
- #define C3H 1385
- #define D3 1468
- #define D3H 1555
- #define E3 1648
- #define F3 1746
- #define F3H 1850
- #define G3 1960
- #define G3H 2076
- #define A3 2200
- #define A3H 2330
- #define B3 2469
- #define C4 2616
- #define C4H 2771
- #define D4 2936
- #define D4H 3111
- #define E4 3296
- #define F4 3492
- #define F4H 3699
- #define G4 3920
- #define G4H 4153
- #define A4 4400
- #define A4H 4661
- #define B4 4938
- #define C5 5232
- #define C5H 5543
- #define D5 5873
- #define D5H 6222

- #define E5 6592
- #define F5 6984
- #define F5H 7399
- #define G5 7839
- #define G5H 8306
- #define A5 8800
- #define A5H 9323
- #define B5 9877
- #define C6 10465
- #define C6H 11087
- #define D6 11746
- #define D6H 12445
- #define E6 13185
- #define F6 13969
- #define F6H 14799
- #define G6 15679
- #define G6H 16612
- #define A6 17600
- #define A6H 18646
- #define B6 19755
- #define C7 20930
- #define C7H 22174
- #define D7 23493
- #define D7H 24890
- #define E7 26370
- #define F7 27938
- #define F7H 29599
- #define G7 31359
- #define G7H 33224
- #define A7 35200
- #define A7H 37293
- #define B7 39510
- #define C8 41860
- #define C8H 44349
- #define D8 46986
- #define D8H 49780
- #define E8 52740
- #define F8 55876
- #define F8H 59199
- #define G8 62719
- #define G8H 66448
- #define A8 70400
- #define A8H 74586
- #define B8 79021

Typedefs

- typedef struct note note_t

4.31.1 Macro Definition Documentation

4.31.1.1 A0

```
#define A0 275
```

4.31.1.2 A0H

```
#define A0H 291
```

4.31.1.3 A1

```
#define A1 550
```

4.31.1.4 A1H

```
#define A1H 582
```

4.31.1.5 A2

```
#define A2 1100
```

4.31.1.6 A2H

```
#define A2H 1165
```

4.31.1.7 A3

```
#define A3 2200
```

4.31.1.8 A3H

```
#define A3H 2330
```

4.31.1.9 A4

```
#define A4 4400
```

4.31.1.10 A4H

```
#define A4H 4661
```

4.31.1.11 A5

```
#define A5 8800
```

4.31.1.12 A5H

```
#define A5H 9323
```

4.31.1.13 A6

```
#define A6 17600
```

4.31.1.14 A6H

```
#define A6H 18646
```

4.31.1.15 A7

```
#define A7 35200
```

4.31.1.16 A7H

```
#define A7H 37293
```

4.31.1.17 A8

```
#define A8 70400
```

4.31.1.18 A8H

```
#define A8H 74586
```

4.31.1.19 B0

```
#define B0 308
```

4.31.1.20 B1

```
#define B1 617
```

4.31.1.21 B2

```
#define B2 1234
```

4.31.1.22 B3

```
#define B3 2469
```

4.31.1.23 B4

```
#define B4 4938
```

4.31.1.24 B5

```
#define B5 9877
```

4.31.1.25 B6

```
#define B6 19755
```

4.31.1.26 B7

```
#define B7 39510
```

4.31.1.27 B8

```
#define B8 79021
```

4.31.1.28 C0

```
#define C0 163
```

4.31.1.29 C0H

```
#define C0H 173
```

4.31.1.30 C1

```
#define C1 327
```

4.31.1.31 C1H

```
#define C1H 346
```

4.31.1.32 C2

```
#define C2 654
```

4.31.1.33 C2H

```
#define C2H 693
```

4.31.1.34 C3

```
#define C3 1308
```

4.31.1.35 C3H

```
#define C3H 1385
```

4.31.1.36 C4

```
#define C4 2616
```

4.31.1.37 C4H

```
#define C4H 2771
```

4.31.1.38 C5

```
#define C5 5232
```

4.31.1.39 C5H

```
#define C5H 5543
```

4.31.1.40 C6

```
#define C6 10465
```

4.31.1.41 C6H

```
#define C6H 11087
```

4.31.1.42 C7

```
#define C7 20930
```

4.31.1.43 C7H

```
#define C7H 22174
```

4.31.1.44 C8

```
#define C8 41860
```

4.31.1.45 C8H

```
#define C8H 44349
```

4.31.1.46 D0

```
#define D0 183
```

4.31.1.47 D0H

```
#define D0H 194
```

4.31.1.48 D1

```
#define D1 367
```

4.31.1.49 D1H

```
#define D1H 388
```

4.31.1.50 D2

```
#define D2 734
```

4.31.1.51 D2H

```
#define D2H 777
```

4.31.1.52 D3

```
#define D3 1468
```

4.31.1.53 D3H

```
#define D3H 1555
```

4.31.1.54 D4

```
#define D4 2936
```

4.31.1.55 D4H

```
#define D4H 3111
```

4.31.1.56 D5

```
#define D5 5873
```

4.31.1.57 D5H

```
#define D5H 6222
```

4.31.1.58 D6

```
#define D6 11746
```

4.31.1.59 D6H

```
#define D6H 12445
```

4.31.1.60 D7

```
#define D7 23493
```

4.31.1.61 D7H

```
#define D7H 24890
```

4.31.1.62 D8

```
#define D8 46986
```

4.31.1.63 D8H

```
#define D8H 49780
```

4.31.1.64 E0

```
#define E0 206
```

4.31.1.65 E1

```
#define E1 412
```

4.31.1.66 E2

```
#define E2 824
```

4.31.1.67 E3

```
#define E3 1648
```

4.31.1.68 E4

```
#define E4 3296
```

4.31.1.69 E5

```
#define E5 6592
```

4.31.1.70 E6

```
#define E6 13185
```

4.31.1.71 E7

```
#define E7 26370
```

4.31.1.72 E8

```
#define E8 52740
```

4.31.1.73 F0

```
#define F0 218
```

4.31.1.74 F0H

```
#define F0H 231
```

4.31.1.75 F1

```
#define F1 436
```

4.31.1.76 F1H

```
#define F1H 462
```

4.31.1.77 F2

```
#define F2 873
```

4.31.1.78 F2H

```
#define F2H 925
```

4.31.1.79 F3

```
#define F3 1746
```

4.31.1.80 F3H

```
#define F3H 1850
```

4.31.1.81 F4

```
#define F4 3492
```

4.31.1.82 F4H

```
#define F4H 3699
```

4.31.1.83 F5

```
#define F5 6984
```

4.31.1.84 F5H

```
#define F5H 7399
```

4.31.1.85 F6

```
#define F6 13969
```

4.31.1.86 F6H

```
#define F6H 14799
```

4.31.1.87 F7

```
#define F7 27938
```

4.31.1.88 F7H

```
#define F7H 29599
```

4.31.1.89 F8

```
#define F8 55876
```

4.31.1.90 F8H

```
#define F8H 59199
```

4.31.1.91 G0

```
#define G0 245
```

4.31.1.92 G0H

```
#define G0H 259
```

4.31.1.93 G1

```
#define G1 490
```

4.31.1.94 G1H

```
#define G1H 519
```

4.31.1.95 G2

```
#define G2 980
```

4.31.1.96 G2H

```
#define G2H 1038
```

4.31.1.97 G3

```
#define G3 1960
```

4.31.1.98 G3H

```
#define G3H 2076
```

4.31.1.99 G4

```
#define G4 3920
```

4.31.1.100 G4H

```
#define G4H 4153
```

4.31.1.101 G5

```
#define G5 7839
```

4.31.1.102 G5H

```
#define G5H 8306
```

4.31.1.103 G6

```
#define G6 15679
```

4.31.1.104 G6H

```
#define G6H 16612
```

4.31.1.105 G7

```
#define G7 31359
```

4.31.1.106 G7H

```
#define G7H 33224
```

4.31.1.107 G8

```
#define G8 62719
```

4.31.1.108 G8H

```
#define G8H 66448
```

4.31.1.109 T1

```
#define T1 4
```

4.31.1.110 T1_1_2

```
#define T1_1_2 6
```

4.31.1.111 T1_2

```
#define T1_2 2
```

4.31.1.112 T1_4

```
#define T1_4 1
```

4.31.1.113 T2

```
#define T2 8
```

4.31.1.114 T4

```
#define T4 16
```

4.31.2 Typedef Documentation

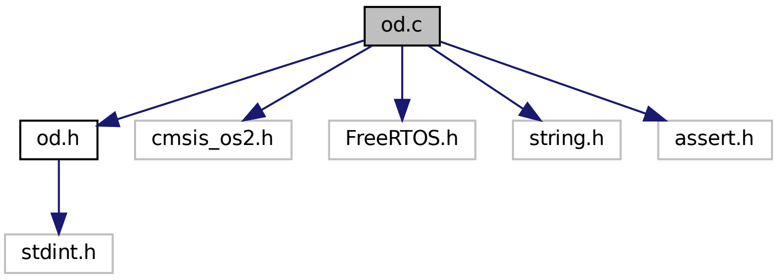
4.31.2.1 note_t

```
typedef struct note note_t
```

4.32 od.c File Reference

```
#include "od.h"
#include <cmsis_os2.h>
#include <FreeRTOS.h>
#include <string.h>
#include <assert.h>
```

Include dependency graph for od.c:



Data Structures

- struct od_entry_t
- struct od_frame_t

Macros

- #define OD_MSGQ_SIZE (16)
- #define DEBUG_NO_CAN 0
- #define ALLOCATE_OD_ENTRY(NAME, ID, TYPE)
- #define LINK_OD_ENTRY(NAME) [(NAME)] = (NAME ## _entry)

Functions

- static void od_unsafe_read (uint8_t data_id, uint8_t ∗dst)
- static void od_unsafe_write (uint8_t data_id, uint8_t ∗src)
- void od_init ()
- void od_update_task (__attribute__((unused)) void ∗argument)

Variables

- static const od_entry_t od_entries [OD_MAX_DATAID]
- osMessageQueueId_t out_q
- osMessageQueueId_t in_q

4.32.1 Macro Definition Documentation

4.32.1.1 ALLOCATE_OD_ENTRY

```
#define ALLOCATE_OD_ENTRY(
                NAME,
                ID,
                TYPE )
```

Value:

```
    enum { NAME = ID }; \
    static_assert((NAME) < OD_MAX_DATAID); \
    static_assert(sizeof(TYPE) < OD_FRAME_MAX_SIZE); \
    static TYPE (NAME ## _var); \
    static const od_entry_t (NAME ## _entry) = { .data_id=(NAME), .size=sizeof(TYPE), .data=(uint8_t*)&(NAME
       ## _var) }; \
    \
    void od_read_ ## NAME  (TYPE *dst) { od_unsafe_read ((NAME), (uint8_t*) dst); } \
    void od_write_ ## NAME (TYPE *src) { od_unsafe_write((NAME), (uint8_t*) src); }
```

4.32.1.2 DEBUG_NO_CAN

```
#define DEBUG_NO_CAN 0
```

4.32.1.3 LINK_OD_ENTRY

```
#define LINK_OD_ENTRY(
              NAME ) [(NAME)] = (NAME ## _entry)
```

4.32.1.4 OD_MSGQ_SIZE

```
#define OD_MSGQ_SIZE (16)
```

4.32.2 Function Documentation

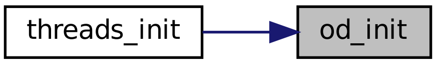
4.32.2.1 od_init()

```
void od_init ( )
```

References in_q, OD_MSGQ_SIZE, and out_q.

Referenced by threads_init().

Here is the caller graph for this function:



4.32.2.2 od_unsafe_read()

```
static void od_unsafe_read (
             uint8_t data_id,
             uint8_t * dst )  [static]
```

Read/write interface

References od_entry_t::data, od_entries, and od_entry_t::size.

4.32.2.3 od_unsafe_write()

```
static void od_unsafe_write (
             uint8_t data_id,
             uint8_t * src )  [static]
```

References od_frame_t::data, od_entry_t::data_id, od_frame_t::data_id, od_entries, out_q, od_entry_t::size, and od_frame_t::size.

4.32.2.4 od_update_task()

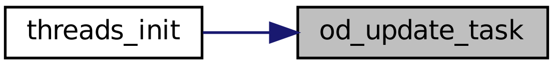
```
void od_update_task (
             __attribute__((unused)) void * argument )
```

Task definition

References od_entry_t::data, od_frame_t::data, od_frame_t::data_id, in_q, od_entries, out_q, and od_entry_t::size.

Referenced by threads_init().

Here is the caller graph for this function:



4.32.3 Variable Documentation

4.32.3.1 in_q

```
osMessageQueueId_t in_q
```

Referenced by od_init(), and od_update_task().

4.32.3.2 od_entries

```
const od_entry_t od_entries[OD_MAX_DATAID]  [static]
```

Initial value:

```
= {
    LINK_OD_ENTRY(TEMPERATURE),
}
```

Object dictionary entries The object dictionary

Referenced by od_unsafe_read(), od_unsafe_write(), and od_update_task().

4.32.3.3 out_q

```
osMessageQueueId_t out_q
```

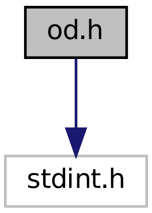
Synchronization primitives

Referenced by od_init(), od_unsafe_write(), and od_update_task().

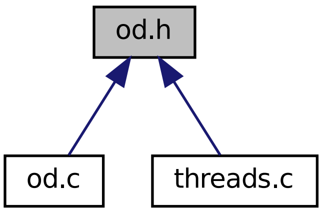
4.33 od.h File Reference

```
#include <stdint.h>
```

Include dependency graph for od.h:



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



Macros

- #define OD_FRAME_MAX_SIZE (64)
- #define OD_MAX_DATAID (256U)
- #define DECLARE_OD_ENTRY(NAME, TYPE)

Functions

- void od_init ()
- void od_update_task (void ∗argument)

4.33.1 Macro Definition Documentation

4.33.1.1 DECLARE_OD_ENTRY

```
#define DECLARE_OD_ENTRY(
            NAME,
            TYPE )
```

Value:

```
    void od_read_ ## NAME  (TYPE *dst); \
    void od_write_ ## NAME (TYPE *src);
```

4.33.1.2 OD_FRAME_MAX_SIZE

```
#define OD_FRAME_MAX_SIZE (64)
```

4.33.1.3 OD_MAX_DATAID

```
#define OD_MAX_DATAID (256U)
```

4.33.2 Function Documentation

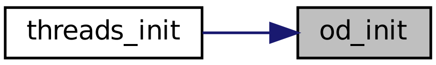
4.33.2.1 od_init()

```
void od_init ( )
```

References in_q, OD_MSGQ_SIZE, and out_q.

Referenced by threads_init().

Here is the caller graph for this function:



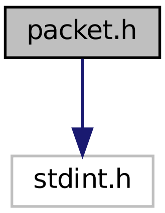
4.33.2.2 od_update_task()

```
void od_update_task (
             void * argument )
```

4.34 packet.h File Reference

```
#include <stdint.h>
```

Include dependency graph for packet.h:



Data Structures

- struct packet_def

Typedefs

- typedef struct packet_def packet_def_t

Variables

- const packet_def_t ping = {0x00, 0x02}

4.34.1 Typedef Documentation

4.34.1.1 packet_def_t

```
typedef struct packet_def packet_def_t
```

4.34.2 Variable Documentation

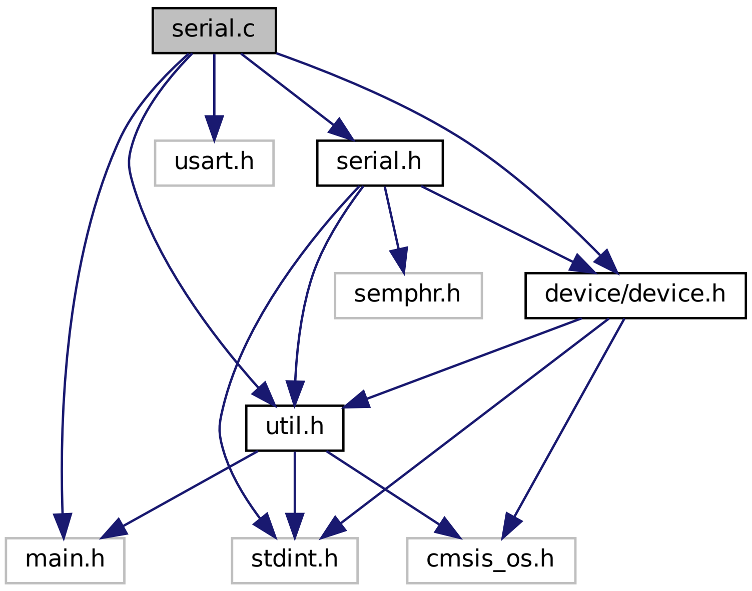
4.34.2.1 ping

```
const packet_def_t ping = {0x00, 0x02}
```

4.35 serial.c File Reference

```
#include <util.h>
#include <usart.h>
#include <main.h>
#include "serial.h"
```

`#include <device/device.h>`
Include dependency graph for serial.c:



Macros

- #define S1_UART huart2
- #define S2_UART huart3
- #define S3_UART huart6
- #define SERIAL_DMA_LEN 32

Functions

- util_error_t serial_data_ready (void ∗context)
- util_error_t serial_send (void ∗context, uint8_t ∗data, uint32_t len)
- util_error_t serial_recv (void ∗context, uint8_t ∗data, uint32_t ∗len)
- util_error_t serial_handle_data (void ∗if_context, void ∗dem_context)
- util_error_t serial_setup_reception (serial_interface_context_t ∗interface_context, serial_transfer_mode_t mode)
- void HAL_UART_RxCpltCallback (UART_HandleTypeDef ∗huart)
- device_deamon_t ∗ serial_get_deamon (void)
- device_interface_t ∗ serial_get_feedback_interface (void)
- util_error_t serial_init (void)
- util_error_t serial_feedback_init (void)

Variables

- static device_deamon_t serial_deamon
- static device_interface_t feedback_interface
- static serial_deamon_context_t serial_deamon_context
- static serial_interface_context_t feedback_interface_context

4.35.1 Macro Definition Documentation

4.35.1.1 S1_UART

```
#define S1_UART huart2
```

4.35.1.2 S2_UART

```
#define S2_UART huart3
```

4.35.1.3 S3_UART

```
#define S3_UART huart6
```

4.35.1.4 SERIAL_DMA_LEN

```
#define SERIAL_DMA_LEN 32
```

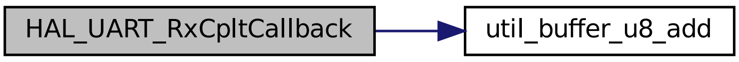
4.35.2 Function Documentation

4.35.2.1 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
             UART_HandleTypeDef * huart )
```

References device_interface::context, device_deamon::context, device_deamon::interfaces, device_deamon↩ ::interfaces_count, serial_interface_context::rx_buffer, serial_interface_context::rx_fragment, serial_deamon_↩ context::rx_sem, serial_deamon, serial_interface_context::uart, and util_buffer_u8_add().

Here is the call graph for this function:



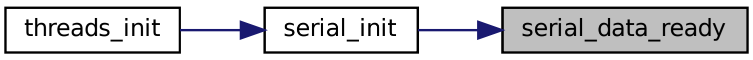
4.35.2.2 serial_data_ready()

```
util_error_t serial_data_ready (
             void * context )
```

References ER_SUCCESS, ER_TIMEOUT, and serial_deamon_context::rx_sem.

Referenced by serial_init().

Here is the caller graph for this function:

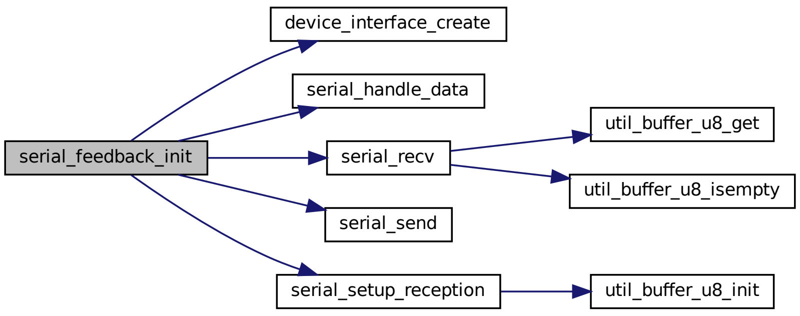


4.35.2.3 serial_feedback_init()

```
util_error_t serial_feedback_init (
             void )
```

References device_interface_create(), ER_SUCCESS, feedback_interface, feedback_interface_context, serial_↩ deamon, serial_handle_data(), serial_recv(), serial_send(), serial_setup_reception(), and SERIAL_TRANSFER_IT.

Here is the call graph for this function:



4.35.2.4 serial_get_deamon()

```
device_deamon_t* serial_get_deamon (
            void )
```

References serial_deamon.

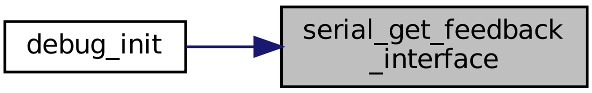
4.35.2.5 serial_get_feedback_interface()

```
device_interface_t* serial_get_feedback_interface (
            void )
```

References feedback_interface.

Referenced by debug_init().

Here is the caller graph for this function:



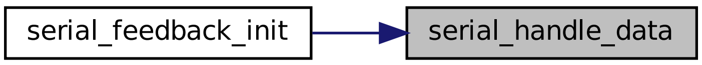
4.35.2.6 serial_handle_data()

```
util_error_t serial_handle_data (
            void * if_context,
            void * dem_context )
```

References ER_SUCCESS.

Referenced by serial_feedback_init().

Here is the caller graph for this function:



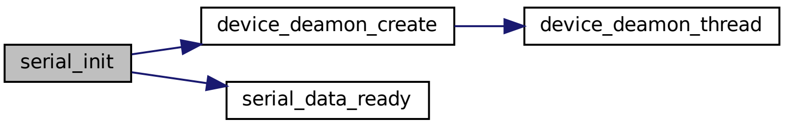
4.35.2.7 serial_init()

```
util_error_t serial_init (
             void  )
```

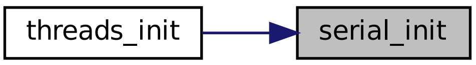
References device_deamon_create(), ER_SUCCESS, serial_deamon_context::rx_sem, serial_deamon_context↩
::rx_sem_buffer, serial_data_ready(), and serial_deamon.

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



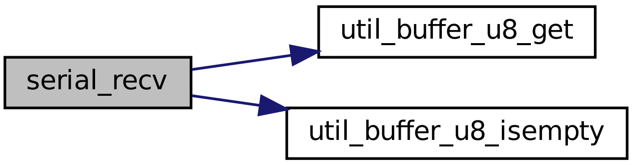
4.35.2.8 serial_recv()

```
util_error_t serial_recv (
             void * context,
             uint8_t * data,
             uint32_t * len )
```

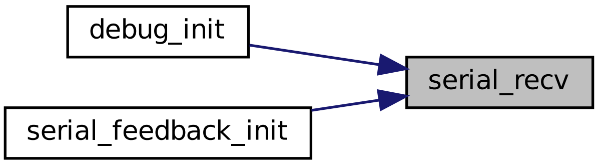
References ER_SUCCESS, serial_interface_context::rx_buffer, util_buffer_u8_get(), and util_buffer_u8_isempty().

Referenced by debug_init(), and serial_feedback_init().

Here is the call graph for this function:



Here is the caller graph for this function:



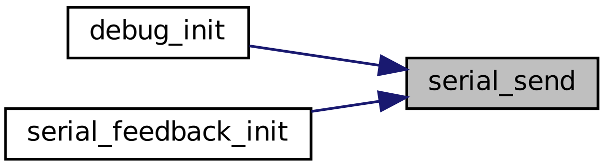
4.35.2.9 serial_send()

```
util_error_t serial_send (
             void * context,
             uint8_t * data,
             uint32_t len )
```

References ER_SUCCESS, and serial_interface_context::uart.

Referenced by debug_init(), and serial_feedback_init().

Here is the caller graph for this function:



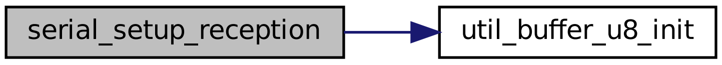
4.35.2.10 serial_setup_reception()

```
util_error_t serial_setup_reception (
             serial_interface_context_t * interface_context,
             serial_transfer_mode_t mode )
```

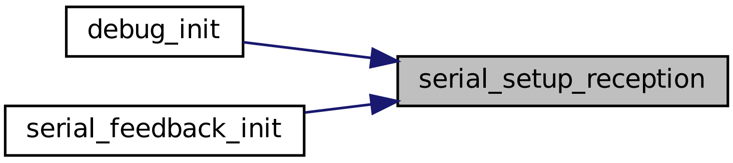
References ER_FAILURE, ER_RESSOURCE_ERROR, serial_interface_context::rx_buffer, serial_interface←_context::rx_data, serial_interface_context::rx_fragment, SERIAL_BUFFER_LEN, SERIAL_TRANSFER_DMA, SERIAL_TRANSFER_IT, serial_interface_context::uart, and util_buffer_u8_init().

Referenced by debug_init(), and serial_feedback_init().

Here is the call graph for this function:



Here is the caller graph for this function:



4.35.3 Variable Documentation

4.35.3.1 feedback_interface

```
device_interface_t feedback_interface  [static]
```

Referenced by debug_init(), serial_feedback_init(), and serial_get_feedback_interface().

4.35.3.2 feedback_interface_context

```
serial_interface_context_t feedback_interface_context [static]
```

Initial value:

```
= {
        .uart = &S3_UART
}
```

Referenced by serial_feedback_init().

4.35.3.3 serial_deamon

```
device_deamon_t serial_deamon [static]
```

Referenced by debug_init(), HAL_UART_RxCpltCallback(), serial_feedback_init(), serial_get_deamon(), and serial_init().

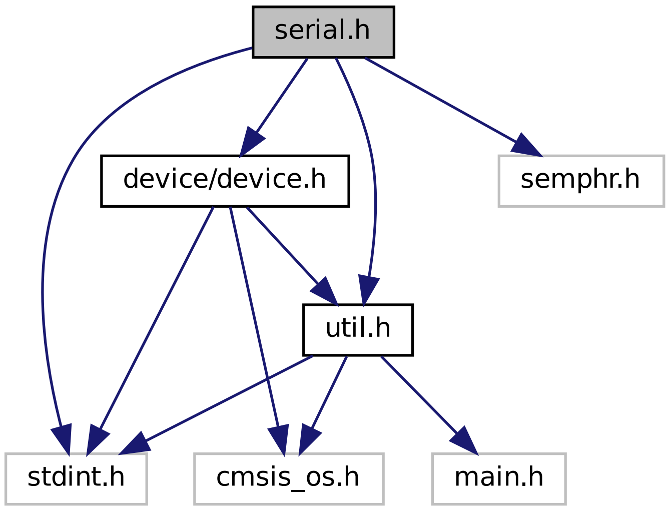
4.35.3.4 serial_deamon_context

```
serial_deamon_context_t serial_deamon_context [static]
```

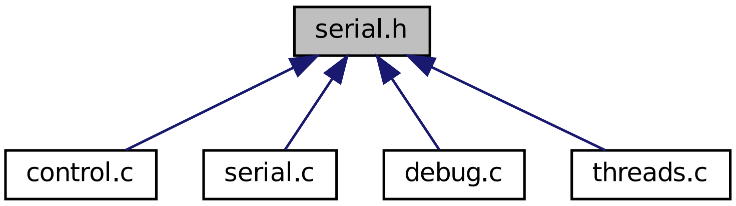
4.36 serial.h File Reference

```
#include <stdint.h>
#include <device/device.h>
#include <util.h>
#include <semphr.h>
```

Include dependency graph for serial.h:



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



Data Structures

- struct serial_deamon_context
- struct serial_interface_context

Macros

- #define SERIAL_BUFFER_LEN 256

Typedefs

- typedef enum serial_interrupt_source serial_interrupt_source_t
- typedef enum serial_transfer_mode serial_transfer_mode_t
- typedef struct serial_deamon_context serial_deamon_context_t
- typedef struct serial_interface_context serial_interface_context_t

Enumerations

- enum serial_interrupt_source { SERIAL_SOURCE_DMA_FIRST_HALF , SERIAL_SOURCE_DMA_SECOND_HALF , SERIAL_SOURCE_IDLE }
- enum serial_transfer_mode { SERIAL_TRANSFER_DMA , SERIAL_TRANSFER_IT }

Functions

- util_error_t serial_init (void)
- util_error_t serial_feedback_init (void)
- device_deamon_t ∗ serial_get_deamon (void)
- device_interface_t ∗ serial_get_feedback_interface (void)
- util_error_t serial_send (void ∗context, uint8_t ∗data, uint32_t len)
- util_error_t serial_recv (void ∗context, uint8_t ∗data, uint32_t ∗len)

4.36.1 Macro Definition Documentation

4.36.1.1 SERIAL_BUFFER_LEN

```
#define SERIAL_BUFFER_LEN 256
```

4.36.2 Typedef Documentation

4.36.2.1 serial_deamon_context_t

```
typedef struct serial_deamon_context serial_deamon_context_t
```

4.36.2.2 serial_interface_context_t

```
typedef struct serial_interface_context serial_interface_context_t
```

4.36.2.3 serial_interrupt_source_t

```
typedef enum serial_interrupt_source serial_interrupt_source_t
```

4.36.2.4 serial_transfer_mode_t

```
typedef enum serial_transfer_mode serial_transfer_mode_t
```

4.36.3 Enumeration Type Documentation

4.36.3.1 serial_interrupt_source

```
enum serial_interrupt_source
```

Enumerator

SERIAL_SOURCE_DMA_FIRST_HALF	
SERIAL_SOURCE_DMA_SECOND_HALF	
SERIAL_SOURCE_IDLE	

4.36.3.2 serial_transfer_mode

```
enum serial_transfer_mode
```

Enumerator

SERIAL_TRANSFER_DMA	
SERIAL_TRANSFER_IT	

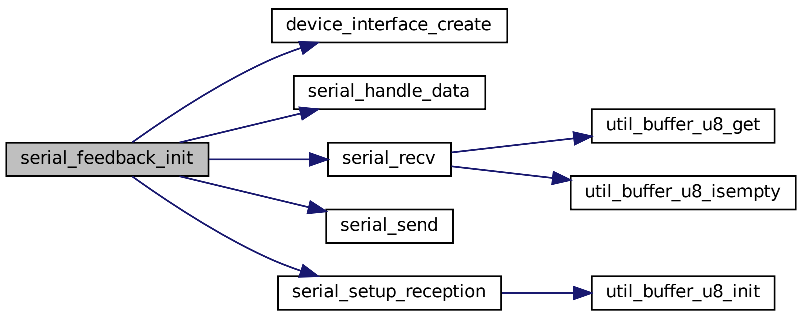
4.36.4 Function Documentation

4.36.4.1 serial_feedback_init()

```
util_error_t serial_feedback_init (
             void  )
```

References device_interface_create(), ER_SUCCESS, feedback_interface, feedback_interface_context, serial_↩ deamon, serial_handle_data(), serial_recv(), serial_send(), serial_setup_reception(), and SERIAL_TRANSFER_IT.

Here is the call graph for this function:



4.36.4.2 serial_get_deamon()

```
device_deamon_t* serial_get_deamon (
             void  )
```

References serial_deamon.

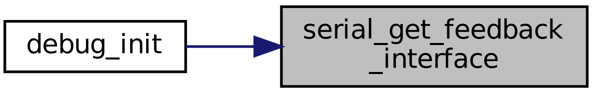
4.36.4.3 serial_get_feedback_interface()

```
device_interface_t* serial_get_feedback_interface (
            void  )
```

References feedback_interface.

Referenced by debug_init().

Here is the caller graph for this function:



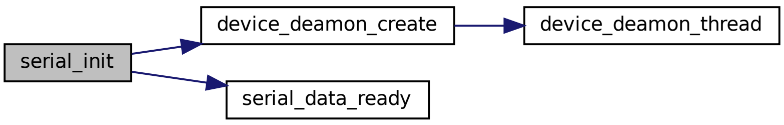
4.36.4.4 serial_init()

```
util_error_t serial_init (
            void  )
```

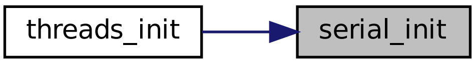
References device_deamon_create(), ER_SUCCESS, serial_deamon_context::rx_sem, serial_deamon_context↩
::rx_sem_buffer, serial_data_ready(), and serial_deamon.

Referenced by threads_init().

Here is the call graph for this function:



Here is the caller graph for this function:



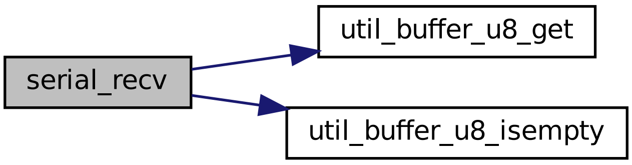
4.36.4.5 serial_recv()

```
util_error_t serial_recv (
             void * context,
             uint8_t * data,
             uint32_t * len )
```

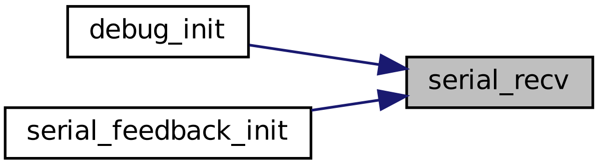
References ER_SUCCESS, serial_interface_context::rx_buffer, util_buffer_u8_get(), and util_buffer_u8_isempty().

Referenced by debug_init(), and serial_feedback_init().

Here is the call graph for this function:



Here is the caller graph for this function:



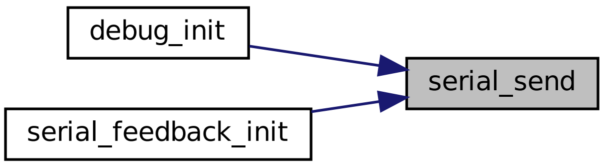
4.36.4.6 serial_send()

```
util_error_t serial_send (
             void * context,
             uint8_t * data,
             uint32_t len )
```

References ER_SUCCESS, and serial_interface_context::uart.

Referenced by debug_init(), and serial_feedback_init().

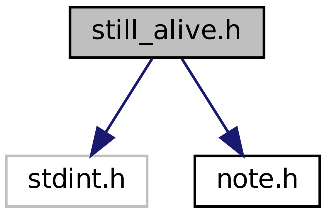
Here is the caller graph for this function:



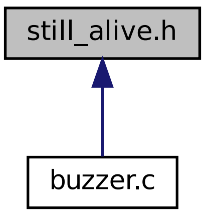
4.37 still_alive.h File Reference

```
#include <stdint.h>
#include "note.h"
```

Include dependency graph for still_alive.h:



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



Variables

- uint16_t still_alive [ ]
- uint32_t still_alive_len = sizeof(still_alive)/sizeof(uint16_t)

4.37.1 Variable Documentation

4.37.1.1 still_alive

```
uint16_t still_alive[ ]
```

Referenced by buzzer_rytm_interrupt().

4.37.1.2 still_alive_len

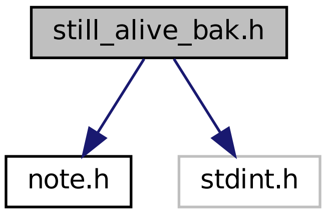
```
uint32_t still_alive_len = sizeof(still_alive)/sizeof(uint16_t)
```

Referenced by buzzer_rytm_interrupt().

4.38 still_alive_bak.h File Reference

```
#include "note.h"
#include <stdint.h>
```

Include dependency graph for still_alive_bak.h:



Variables

- note_t still_alive [ ]
- uint32_t still_alive_len = sizeof(still_alive)/sizeof(note_t)

4.38.1 Variable Documentation

4.38.1.1 still_alive

```
note_t still_alive[]
```

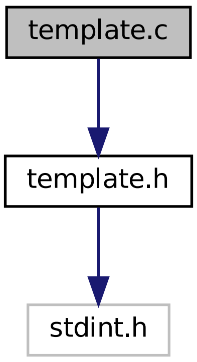
4.38.1.2 still_alive_len

```
uint32_t still_alive_len = sizeof(still_alive)/sizeof(note_t)
```

4.39 template.c File Reference

```
#include "template.h"
```

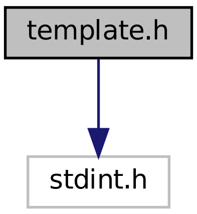
Include dependency graph for template.c:



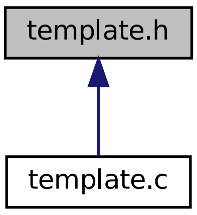
4.40 template.h File Reference

```
#include <stdint.h>
```

Include dependency graph for template.h:



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

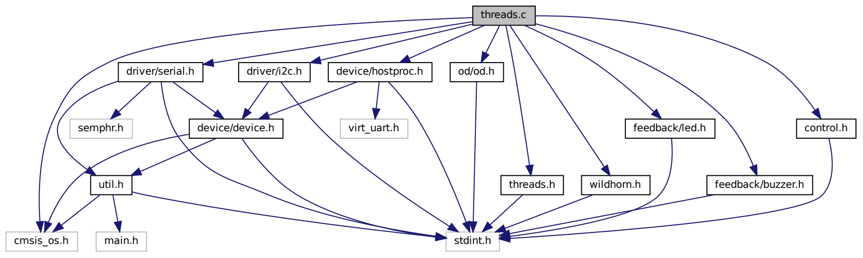


4.41 threads.c File Reference

```
#include <cmsis_os.h>
#include <threads.h>
#include <wildhorn.h>
#include <feedback/led.h>
#include <feedback/buzzer.h>
#include <driver/serial.h>
#include <driver/i2c.h>
#include <control.h>
#include <device/hostproc.h>
```

`#include <od/od.h>`

Include dependency graph for threads.c:



Macros

- #define DEFAULT_SZ (1024)
- #define OD_SZ DEFAULT_SZ
- #define OD_PRIO (6)
- #define CONTROL_SZ DEFAULT_SZ
- #define CONTROL_PRIO (6)
- #define LED_RGB_SZ DEFAULT_SZ
- #define LED_RGB_PRIO (0)
- #define CREATE_THREAD(handle, name, func, cont, sz, prio)

  macro to declare a static thread in FreeRTOS

Functions

- void threads_init (void)

  Initialize all the threads of Wildhorn AV.

Variables

- static TaskHandle_t od_handle = NULL
- static TaskHandle_t control_handle = NULL
- static TaskHandle_t led_rgb_handle = NULL

4.41.1 Macro Definition Documentation

4.41.1.1 CONTROL_PRIO

```
#define CONTROL_PRIO (6)
```

4.41.1.2 CONTROL_SZ

```
#define CONTROL_SZ DEFAULT_SZ
```

4.41.1.3 CREATE_THREAD

```
#define CREATE_THREAD(
              handle,
              name,
              func,
              cont,
              sz,
              prio )
```

Value:

```
    static StaticTask_t name##_buffer; \
    static StackType_t name##_stack[ sz ]; \
    handle = xTaskCreateStatic( \
            func, \
            #name, \
            sz, \
            ( void * ) cont, \
            prio, \
            name##_stack, \
            &name##_buffer)
```

macro to declare a static thread in FreeRTOS

This macros make the necessary funtion calls to setup a stack and working area for the declaration of a static FreeRTOS thread.

Parameters

handle	A `TaskHandle_t` object to reference the created Thread.
name	A name for thread.
func	The entry point for the thread.
cont	The context for the thread.
sz	The desired size for the thread stack.
prio	The priority for the thread.

4.41.1.4 DEFAULT_SZ

```
#define DEFAULT_SZ (1024)
```

4.41.1.5 LED_RGB_PRIO

```
#define LED_RGB_PRIO (0)
```

4.41.1.6 LED_RGB_SZ

```
#define LED_RGB_SZ DEFAULT_SZ
```

4.41.1.7 OD_PRIO

```
#define OD_PRIO (6)
```

4.41.1.8 OD_SZ

```
#define OD_SZ DEFAULT_SZ
```

4.41.2 Function Documentation

4.41.2.1 threads_init()

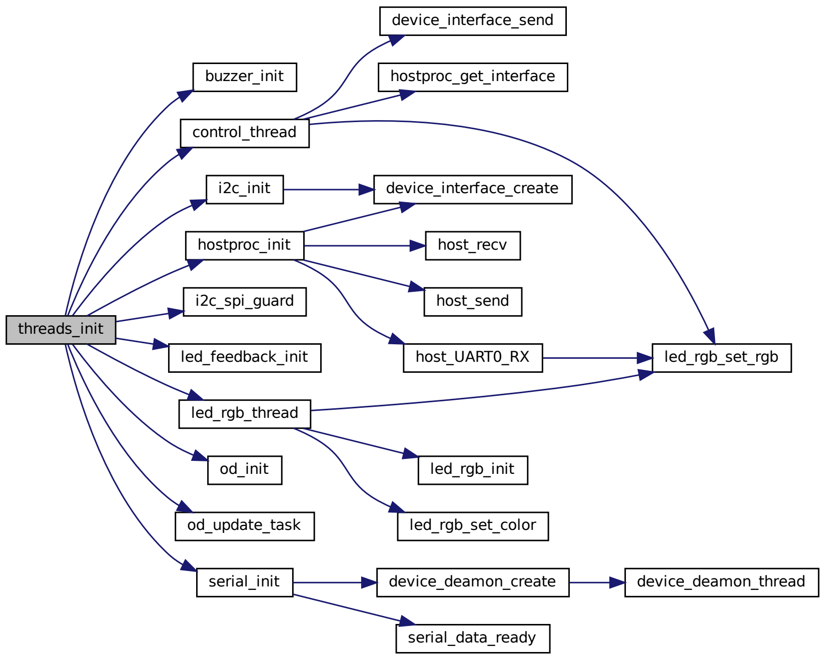
```
void threads_init (
             void  )
```

Initialize all the threads of Wildhorn AV.

This is the only function that needs to be called from the ST Auto-generated files. This is clever in case the autogeneration fails. This will minimize the code to be rewritten.

References buzzer_init(), control_handle, CONTROL_PRIO, CONTROL_SZ, control_thread(), CREATE_THREAD, ER_SUCCESS, hostproc_init(), i2c_init(), i2c_spi_guard(), led_feedback_init(), led_rgb_handle, LED_RGB_PRIO, LED_RGB_SZ, led_rgb_thread(), od_handle, od_init(), OD_PRIO, OD_SZ, od_update_task(), and serial_init().

Here is the call graph for this function:



4.41.3 Variable Documentation

4.41.3.1 control_handle

```
TaskHandle_t control_handle = NULL  [static]
```

Referenced by threads_init().

4.41.3.2 led_rgb_handle

```
TaskHandle_t led_rgb_handle = NULL  [static]
```

Referenced by threads_init().

4.41.3.3 od_handle

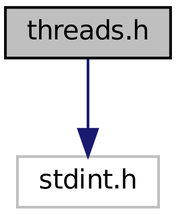
```
TaskHandle_t od_handle = NULL  [static]
```

Referenced by threads_init().

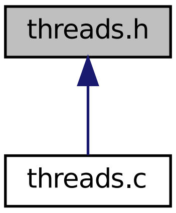
4.42 threads.h File Reference

```
#include <stdint.h>
```

Include dependency graph for threads.h:



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



Functions

- void threads_init (void)

  Initialize all the threads of Wildhorn AV.

4.42.1 Function Documentation

4.42.1.1 threads_init()

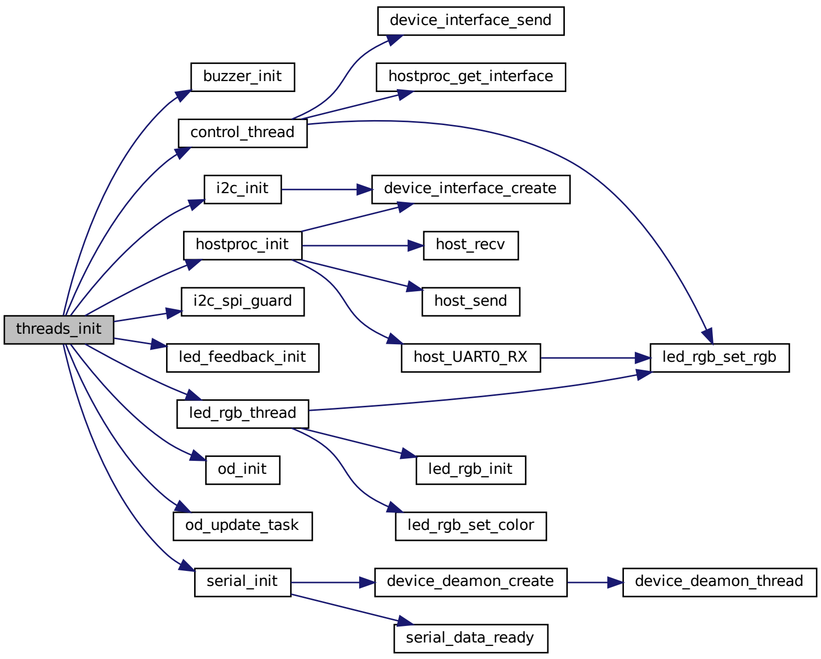
```
void threads_init (
             void  )
```

Initialize all the threads of Wildhorn AV.

This is the only function that needs to be called from the ST Auto-generated files. This is clever in case the autogeneration fails. This will minimize the code to be rewritten.

References buzzer_init(), control_handle, CONTROL_PRIO, CONTROL_SZ, control_thread(), CREATE_THREAD, ER_SUCCESS, hostproc_init(), i2c_init(), i2c_spi_guard(), led_feedback_init(), led_rgb_handle, LED_RGB_PRIO, LED_RGB_SZ, led_rgb_thread(), od_handle, od_init(), OD_PRIO, OD_SZ, od_update_task(), and serial_init().

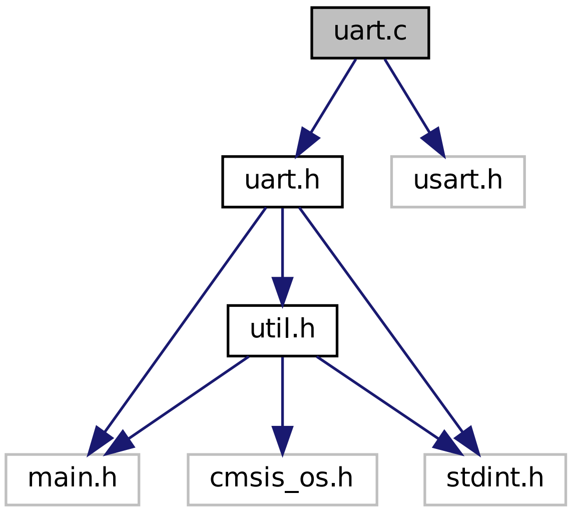
Here is the call graph for this function:



4.43 uart.c File Reference

```
#include "uart.h"
#include <usart.h>
```

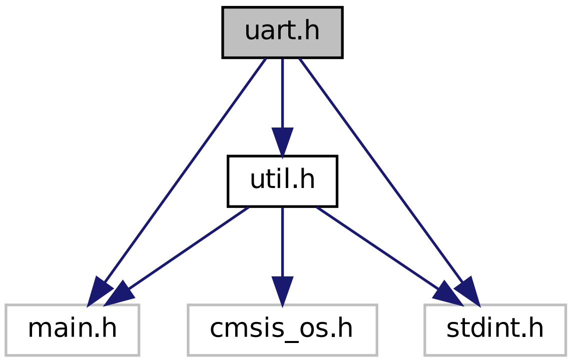
Include dependency graph for uart.c:



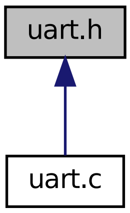
4.44 uart.h File Reference

```
#include <main.h>
#include <stdint.h>
#include <util.h>
```

Include dependency graph for uart.h:



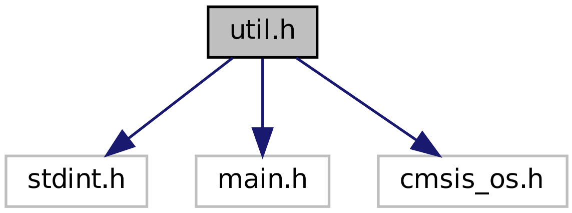
This graph shows which files directly or indirectly include this file:



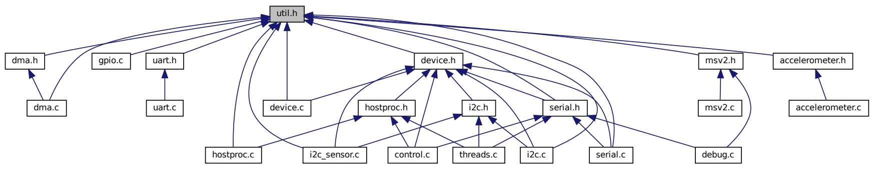
4.45 util.h File Reference

```
#include <stdint.h>
#include <main.h>
#include <cmsis_os.h>
```

Include dependency graph for util.h:



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



Data Structures

- struct util_buffer_u8
- struct util_buffer_u16
- struct util_buffer_i16

Macros

- #define WRITE_IN_REG(reg, mask, data) (reg) &= ∼(mask); (reg) |= (data)

  Macro to write masked data into a register.
- #define ENTER_CRITICAL taskENTER_CRITICAL

  Macro to enter a critical section.
- #define EXIT_CRITICAL taskEXIT_CRITICAL

  Macro to exit a critical section.
- #define UTIL_GENERATE_BUFFER(type, name)
- #define util_abs(a) ((a)<0?-(a):(a))

Typedefs

- typedef enum util_error util_error_t

  Unified error codes for the whole WildhornAV project.
- typedef struct util_buffer_u8 util_buffer_u8_t
- typedef struct util_buffer_u16 util_buffer_u16_t
- typedef struct util_buffer_i16 util_buffer_i16_t

Enumerations

- enum util_error {
  ER_SUCCESS = 0 , ER_DATA_NOT_RDY = 0<<1 , ER_FAILURE = 1<<1 , ER_OUT_OF_RANGE = 1<<2
  ,
  ER_TIMEOUT = 1<<3 , ER_RESSOURCE_ERROR = 1<<4 }

  Unified error codes for the whole WildhornAV project.

Functions

- static void util_encode_u8 (uint8_t ∗data, uint8_t value)
- static void util_encode_u16 (uint8_t ∗data, uint16_t value)
- static void util_encode_u32 (uint8_t ∗data, uint32_t value)
- static void util_encode_i8 (uint8_t ∗data, int8_t value)
- static void util_encode_i16 (uint8_t ∗data, int16_t value)
- static void util_encode_i32 (uint8_t ∗data, int32_t value)
- static uint8_t util_decode_u8 (uint8_t ∗data)
- static uint16_t util_decode_u16 (uint8_t ∗data)
- static uint32_t util_decode_u32 (uint8_t ∗data)
- static int8_t util_decode_i8 (uint8_t ∗data)
- static int16_t util_decode_i16 (uint8_t ∗data)
- static int32_t util_decode_i32 (uint8_t ∗data)
- static void util_buffer_u8_init (util_buffer_u8_t ∗bfr, uint8_t ∗buffer, uint16_t bfr_len)
- static void util_buffer_u8_add (util_buffer_u8_t ∗bfr, uint8_t d)
- static uint8_t util_buffer_u8_get (util_buffer_u8_t ∗bfr)
- static uint8_t util_buffer_u8_access (util_buffer_u8_t ∗bfr, int16_t ix)
- static uint8_t util_buffer_u8_isempty (util_buffer_u8_t ∗bfr)
- static void util_buffer_u16_init (util_buffer_u16_t ∗bfr, uint16_t ∗buffer, uint16_t bfr_len)
- static void util_buffer_u16_add (util_buffer_u16_t ∗bfr, uint16_t d)
- static uint16_t util_buffer_u16_get (util_buffer_u16_t ∗bfr)
- static uint8_t util_buffer_u16_isempty (util_buffer_u16_t ∗bfr)
- static void util_buffer_i16_init (util_buffer_i16_t ∗bfr, int16_t ∗buffer, uint16_t bfr_len)
- static void util_buffer_i16_add (util_buffer_i16_t ∗bfr, int16_t d)
- static int16_t util_buffer_i16_get (util_buffer_i16_t ∗bfr)
- static uint8_t util_buffer_i16_isempty (util_buffer_i16_t ∗bfr)

4.45.1 Macro Definition Documentation

4.45.1.1 ENTER_CRITICAL

```
#define ENTER_CRITICAL taskENTER_CRITICAL
```

Macro to enter a critical section.

4.45.1.2 EXIT_CRITICAL

```
#define EXIT_CRITICAL taskEXIT_CRITICAL
```

Macro to exit a critical section.

4.45.1.3 util_abs

```
#define util_abs(
              a ) ((a)<0?-(a):(a))
```

4.45.1.4 UTIL_GENERATE_BUFFER

```
#define UTIL_GENERATE_BUFFER(
              type,
              name )
```

Value:

```
    typedef struct UTIL_BUFFER_##name{
          \
        uint16_t c_ix;
          \
        uint16_t l_ix;
          \
        uint16_t bfr_len;
          \
        type * buffer;
          \
    }UTIL_BUFFER_##name##_t;
          \
static inline void util_buffer_##name##_init(UTIL_BUFFER_##name##_t * bfr, type * buffer, uint16_t bfr_len)
       {   \
    bfr->c_ix = 0;
          \
    bfr->l_ix = 0;
          \
    bfr->bfr_len = bfr_len;
          \
    bfr->buffer = buffer;
          \
}
          \
static inline void util_buffer_##name##_add(UTIL_BUFFER_##name##_t * bfr, type d) {
          \
    bfr->buffer[bfr->c_ix++] = d;
          \
    if(bfr->c_ix == bfr->bfr_len) bfr->c_ix = 0;
          \
}
          \
static inline type util_buffer_##name##_get(UTIL_BUFFER_##name##_t * bfr) {
          \
    type tmp = bfr->buffer[bfr->l_ix++];
          \
    if(bfr->l_ix == bfr->bfr_len) bfr->l_ix=0;
          \
    return tmp;
          \
}
          \
static inline type util_buffer_##name##_access(UTIL_BUFFER_##name##_t * bfr, uint16_t ix) {
          \
    int16_t i = bfr->c_ix - ix - 1;
          \
    while(i < 0) i += bfr->bfr_len;
          \
    return bfr->buffer[i];
          \
}
          \
static inline uint8_t util_buffer_##name##_isempty(UTIL_BUFFER_##name##_t * bfr) {
          \
    return bfr->l_ix == bfr->c_ix;
          \
}
```

4.45.1.5 WRITE_IN_REG

```
#define WRITE_IN_REG(
                reg,
                mask,
                data ) (reg) &= ~(mask); (reg) |= (data)
```

Macro to write masked data into a register.

4.45.2 Typedef Documentation

4.45.2.1 util_buffer_i16_t

```
typedef struct util_buffer_i16 util_buffer_i16_t
```

4.45.2.2 util_buffer_u16_t

```
typedef struct util_buffer_u16 util_buffer_u16_t
```

4.45.2.3 util_buffer_u8_t

```
typedef struct util_buffer_u8 util_buffer_u8_t
```

4.45.2.4 util_error_t

```
typedef enum util_error util_error_t
```

Unified error codes for the whole WildhornAV project.

Note

The error codes can be ORed together to create more complex errors.

4.45.3 Enumeration Type Documentation

4.45.3.1 util_error

```
enum util_error
```

Unified error codes for the whole WildhornAV project.

Note

The error codes can be ORed together to create more complex errors.

Enumerator

ER_SUCCESS	Operation completed successfully
ER_DATA_NOT_RDY	Error due to lack of readiness
ER_FAILURE	Error due to a generic failure
ER_OUT_OF_RANGE	Error due to a range issue
ER_TIMEOUT	Error due to a timeout
ER_RESSOURCE_ERROR	Error due to a ressource issue

4.45.4 Function Documentation

4.45.4.1 util_buffer_i16_add()

```
static void util_buffer_i16_add (
            util_buffer_i16_t * bfr,
             int16_t d )  [inline], [static]
```

References util_buffer_i16::bfr_len, util_buffer_i16::buffer, and util_buffer_i16::c_ix.

4.45.4.2 util_buffer_i16_get()

```
static int16_t util_buffer_i16_get (
             util_buffer_i16_t * bfr )  [inline], [static]
```

References util_buffer_i16::bfr_len, util_buffer_i16::buffer, and util_buffer_i16::l_ix.

4.45.4.3 util_buffer_i16_init()

```
static void util_buffer_i16_init (
             util_buffer_i16_t * bfr,
             int16_t * buffer,
             uint16_t bfr_len )  [inline], [static]
```

References util_buffer_i16::bfr_len, util_buffer_i16::buffer, util_buffer_i16::c_ix, and util_buffer_i16::l_ix.

4.45.4.4 util_buffer_i16_isempty()

```
static uint8_t util_buffer_i16_isempty (
             util_buffer_i16_t * bfr )  [inline], [static]
```

References util_buffer_i16::c_ix, and util_buffer_i16::l_ix.

4.45.4.5 util_buffer_u16_add()

```
static void util_buffer_u16_add (
             util_buffer_u16_t * bfr,
             uint16_t d )  [inline], [static]
```

References util_buffer_u16::bfr_len, util_buffer_u16::buffer, and util_buffer_u16::c_ix.

4.45.4.6 util_buffer_u16_get()

```
static uint16_t util_buffer_u16_get (
             util_buffer_u16_t * bfr )  [inline], [static]
```

References util_buffer_u16::bfr_len, util_buffer_u16::buffer, and util_buffer_u16::l_ix.

4.45.4.7 util_buffer_u16_init()

```
static void util_buffer_u16_init (
             util_buffer_u16_t * bfr,
             uint16_t * buffer,
             uint16_t bfr_len )  [inline], [static]
```

References util_buffer_u16::bfr_len, util_buffer_u16::buffer, util_buffer_u16::c_ix, and util_buffer_u16::l_ix.

4.45.4.8 util_buffer_u16_isempty()

```
static uint8_t util_buffer_u16_isempty (
             util_buffer_u16_t * bfr )  [inline], [static]
```

References util_buffer_u16::c_ix, and util_buffer_u16::l_ix.

4.45.4.9 util_buffer_u8_access()

```
static uint8_t util_buffer_u8_access (
             util_buffer_u8_t * bfr,
             int16_t ix )  [inline], [static]
```

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, and util_buffer_u8::c_ix.

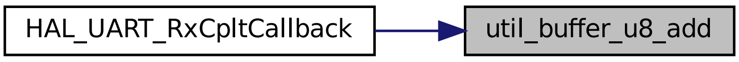
4.45.4.10 util_buffer_u8_add()

```
static void util_buffer_u8_add (
             util_buffer_u8_t * bfr,
             uint8_t d )  [inline], [static]
```

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, and util_buffer_u8::c_ix.

Referenced by HAL_UART_RxCpltCallback().

Here is the caller graph for this function:



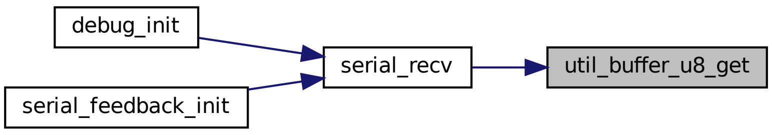
4.45.4.11 util_buffer_u8_get()

```
static uint8_t util_buffer_u8_get (
            util_buffer_u8_t * bfr )  [inline], [static]
```

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, and util_buffer_u8::l_ix.

Referenced by serial_recv().

Here is the caller graph for this function:



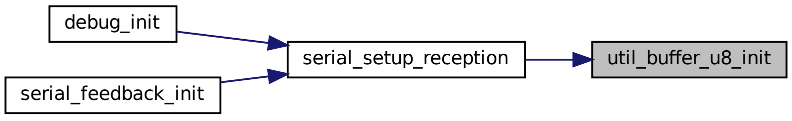
4.45.4.12 util_buffer_u8_init()

```
static void util_buffer_u8_init (
            util_buffer_u8_t * bfr,
            uint8_t * buffer,
            uint16_t bfr_len )  [inline], [static]
```

References util_buffer_u8::bfr_len, util_buffer_u8::buffer, util_buffer_u8::c_ix, and util_buffer_u8::l_ix.

Referenced by serial_setup_reception().

Here is the caller graph for this function:



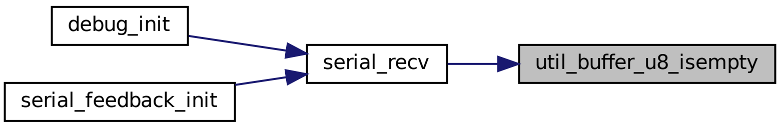
4.45.4.13 util_buffer_u8_isempty()

```
static uint8_t util_buffer_u8_isempty (
             util_buffer_u8_t * bfr )  [inline], [static]
```

References util_buffer_u8::c_ix, and util_buffer_u8::l_ix.

Referenced by serial_recv().

Here is the caller graph for this function:

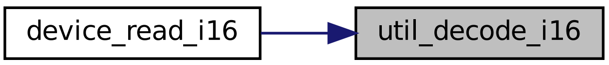


4.45.4.14 util_decode_i16()

```
static int16_t util_decode_i16 (
             uint8_t * data )  [inline], [static]
```

Referenced by device_read_i16().

Here is the caller graph for this function:

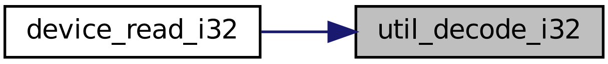


4.45.4.15 util_decode_i32()

```
static int32_t util_decode_i32 (
             uint8_t * data )  [inline], [static]
```

Referenced by device_read_i32().

Here is the caller graph for this function:

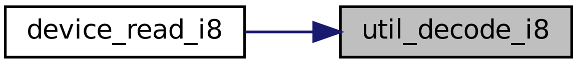


4.45.4.16 util_decode_i8()

```
static int8_t util_decode_i8 (
             uint8_t * data )  [inline], [static]
```

Referenced by device_read_i8().

Here is the caller graph for this function:

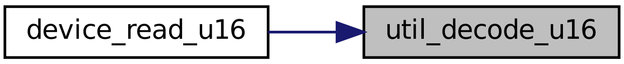


4.45.4.17 util_decode_u16()

```
static uint16_t util_decode_u16 (
             uint8_t * data )  [inline], [static]
```

Referenced by device_read_u16().

Here is the caller graph for this function:

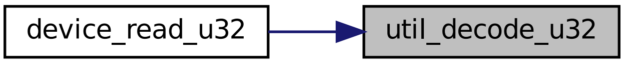


4.45.4.18 util_decode_u32()

```
static uint32_t util_decode_u32 (
             uint8_t * data )  [inline], [static]
```

Referenced by device_read_u32().

Here is the caller graph for this function:

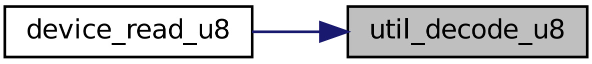


4.45.4.19 util_decode_u8()

```
static uint8_t util_decode_u8 (
             uint8_t * data )  [inline], [static]
```

Referenced by device_read_u8().

Here is the caller graph for this function:

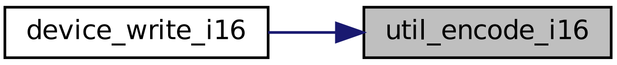


4.45.4.20 util_encode_i16()

```
static void util_encode_i16 (
            uint8_t * data,
            int16_t value )  [inline], [static]
```

Referenced by device_write_i16().

Here is the caller graph for this function:

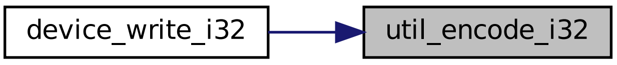


4.45.4.21 util_encode_i32()

```
static void util_encode_i32 (
            uint8_t * data,
            int32_t value )  [inline], [static]
```

Referenced by device_write_i32().

Here is the caller graph for this function:

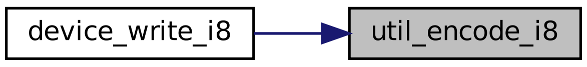


4.45.4.22 util_encode_i8()

```
static void util_encode_i8 (
             uint8_t * data,
             int8_t value )  [inline], [static]
```

Referenced by device_write_i8().

Here is the caller graph for this function:

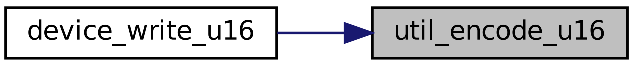


4.45.4.23 util_encode_u16()

```
static void util_encode_u16 (
             uint8_t * data,
             uint16_t value )  [inline], [static]
```

Referenced by device_write_u16().

Here is the caller graph for this function:

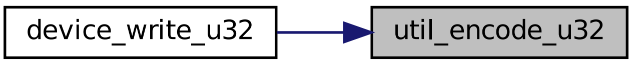


4.45.4.24 util_encode_u32()

```
static void util_encode_u32 (
             uint8_t * data,
             uint32_t value )  [inline], [static]
```

Referenced by device_write_u32().

Here is the caller graph for this function:

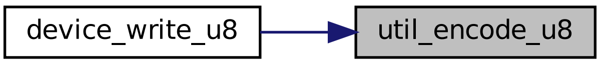


4.45.4.25 util_encode_u8()

```
static void util_encode_u8 (
             uint8_t * data,
             uint8_t value )  [inline], [static]
```

Referenced by device_write_u8().

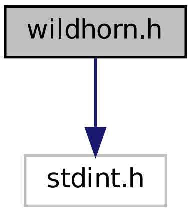
Here is the caller graph for this function:



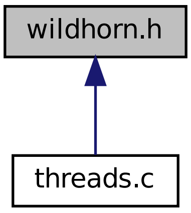
4.46 wildhorn.h File Reference

```
#include <stdint.h>
```

Include dependency graph for wildhorn.h:



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



Macros

- #define WH_TRUE 1
- #define WH_FALSE 0
- #define WH_HAS_SENSORS WH_TRUE
- #define WH_HAS_FEEDBACK WH_TRUE
- #define WH_HAS_RADIO WH_FALSE
- #define WH_HAS_GNSS WH_FALSE
- #define WH_HAS_KRTEK WH_FALSE
- #define WH_USE_BUZZER WH_FALSE

4.46.1 Macro Definition Documentation

4.46.1.1 WH_FALSE

```
#define WH_FALSE 0
```

4.46.1.2 WH_HAS_FEEDBACK

```
#define WH_HAS_FEEDBACK WH_TRUE
```

4.46.1.3 WH_HAS_GNSS

```
#define WH_HAS_GNSS WH_FALSE
```

4.46.1.4 WH_HAS_KRTEK

```
#define WH_HAS_KRTEK WH_FALSE
```

4.46.1.5 WH_HAS_RADIO

```
#define WH_HAS_RADIO WH_FALSE
```

4.46.1.6 WH_HAS_SENSORS

```
#define WH_HAS_SENSORS WH_TRUE
```

4.46.1.7 WH_TRUE

```
#define WH_TRUE 1
```

4.46.1.8 WH_USE_BUZZER

```
#define WH_USE_BUZZER WH_FALSE
```

Index