

Access control system panel firmware

1.0

Generated by Doxygen 1.8.16

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 acs_msg_data_auth_req_t Struct Reference	5
3.2 acs_msg_data_auth_resp_t Struct Reference	5
3.3 acs_msg_data_door_ctrl_t Struct Reference	5
3.4 acs_msg_head_t Union Reference	6
3.5 cache_item_t Union Reference	6
3.5.1 Detailed Description	6
3.6 reader_conf_t Struct Reference	7
3.7 reader_wiring_t Struct Reference	7
3.8 weigand26_buff_item_t Struct Reference	7
3.9 weigand26_frame_t Union Reference	8
3.10 weigand26_t Struct Reference	8
4 File Documentation	9
4.1 acs_can_protocol.h File Reference	9
4.1.1 Detailed Description	10
4.2 brownout.c File Reference	10
4.2.1 Detailed Description	11
4.2.2 Function Documentation	11
4.2.2.1 BOD_Init()	11
4.2.2.2 BOD_IRQHandler()	11
4.3 brownout.h File Reference	11
4.3.1 Detailed Description	12
4.3.2 Function Documentation	12
4.3.2.1 BOD_Init()	12
4.3.2.2 BOD_IRQHandler()	12
4.4 can_term_driver.c File Reference	12
4.4.1 Detailed Description	13
4.4.2 Function Documentation	14
4.4.2.1 _timing_calculate()	14
4.4.2.2 CAN_init()	14
4.4.2.3 CAN_IRQHandler()	14
4.4.2.4 CAN_recv_filter()	14
4.4.2.5 CAN_recv_filter_all_ext()	15
4.4.2.6 CAN_send_once()	15
4.4.2.7 CAN_send_test()	15
4.5 can_term_driver.h File Reference	16

4.5.1 Detailed Description	16
4.5.2 Function Documentation	17
4.5.2.1 CAN_init()	17
4.5.2.2 CAN_IRQHandler()	17
4.5.2.3 CAN_recv_filter()	17
4.5.2.4 CAN_recv_filter_all_ext()	18
4.5.2.5 CAN_send_once()	18
4.5.2.6 CAN_send_test()	18
4.6 reader.c File Reference	19
4.6.1 Detailed Description	20
4.6.2 Function Documentation	20
4.6.2.1 reader_deinit()	20
4.6.2.2 reader_get_request_from_buffer()	20
4.6.2.3 reader_init()	21
4.6.2.4 reader_is_door_open()	21
4.6.2.5 reader_unlock()	21
4.6.3 Variable Documentation	21
4.6.3.1 reader_conf	21
4.7 reader.h File Reference	22
4.7.1 Detailed Description	23
4.7.2 Function Documentation	23
4.7.2.1 reader_deinit()	23
4.7.2.2 reader_get_request_from_buffer()	23
4.7.2.3 reader_init()	24
4.7.2.4 reader_is_door_open()	24
4.7.2.5 reader_unlock()	24
4.8 start.c File Reference	24
4.8.1 Detailed Description	25
4.8.2 Function Documentation	25
4.8.2.1 _check_system_stack_size()	25
4.8.2.2 main()	26
4.9 static_cache.c File Reference	26
4.9.1 Detailed Description	26
4.10 static_cache.h File Reference	26
4.10.1 Detailed Description	27
4.10.2 Macro Definition Documentation	27
4.10.2.1 STATIC_CACHE_SETS	27
4.10.3 Function Documentation	27
4.10.3.1 static_cache_convert()	27
4.10.3.2 static_cache_erase()	28
4.10.3.3 static_cache_get()	28
4.10.3.4 static_cache_insert()	29

4.10.3.5 static_cache_reset()	29
4.11 storage.c File Reference	29
4.11.1 Detailed Description	30
4.11.2 Function Documentation	30
4.11.2.1 I2C_IRQHandler()	30
4.11.2.2 storage_init()	30
4.11.2.3 storage_read_byte()	30
4.11.2.4 storage_read_word_le()	31
4.11.2.5 storage_write_byte()	31
4.11.2.6 storage_write_word_le()	31
4.12 storage.h File Reference	31
4.12.1 Detailed Description	32
4.12.2 Function Documentation	32
4.12.2.1 storage_init()	32
4.12.2.2 storage_read_byte()	33
4.12.2.3 storage_read_word_le()	33
4.12.2.4 storage_write_byte()	33
4.12.2.5 storage_write_word_le()	33
4.13 terminal.c File Reference	34
4.13.1 Detailed Description	35
4.13.2 Function Documentation	35
4.13.2.1 map_reader_idx_to_cache()	35
4.13.2.2 term_can_error()	35
4.13.2.3 term_can_recv()	35
4.13.2.4 term_can_send()	36
4.13.2.5 terminal_init()	36
4.13.2.6 terminal_reconfigure()	36
4.13.3 Variable Documentation	37
4.13.3.1 USER_REQUEST_WAIT_MS	37
4.14 terminal.h File Reference	37
4.14.1 Detailed Description	37
4.14.2 Function Documentation	37
4.14.2.1 term_can_error()	37
4.14.2.2 term_can_recv()	38
4.14.2.3 term_can_send()	38
4.14.2.4 terminal_init()	38
4.14.2.5 terminal_reconfigure()	39
4.15 terminal_config.c File Reference	39
4.15.1 Detailed Description	40
4.15.2 Function Documentation	40
4.15.2.1 get_reader_a_addr()	40
4.15.2.2 get_reader_b_addr()	40

4.15.2.3 set_reader_addr()	40
4.15.2.4 terminal_config_init()	41
4.16 terminal_config.h File Reference	41
4.16.1 Detailed Description	43
4.16.2 Function Documentation	43
4.16.2.1 get_reader_a_addr()	43
4.16.2.2 get_reader_b_addr()	43
4.16.2.3 set_reader_addr()	43
4.16.2.4 terminal_config_init()	44
4.17 watchdog.c File Reference	44
4.17.1 Detailed Description	44
4.17.2 Function Documentation	45
4.17.2.1 WDT_Feed()	45
4.17.2.2 WDT_Init()	45
4.17.2.3 WDT_IRQHandler()	45
4.18 watchdog.h File Reference	45
4.18.1 Detailed Description	46
4.18.2 Function Documentation	46
4.18.2.1 WDT_Feed()	46
4.18.2.2 WDT_Init()	46
4.18.2.3 WDT_IRQHandler()	47
4.19 weigand.c File Reference	47
4.19.1 Detailed Description	48
4.19.2 Function Documentation	48
4.19.2.1 weigand_disable()	48
4.19.2.2 weigand_get_card()	48
4.19.2.3 weigand_get_facility()	49
4.19.2.4 weigand_init()	49
4.19.2.5 weigand_is_parity_ok()	50
4.20 weigand.h File Reference	50
4.20.1 Detailed Description	51
4.20.2 Function Documentation	51
4.20.2.1 weigand_disable()	51
4.20.2.2 weigand_get_card()	51
4.20.2.3 weigand_get_facility()	52
4.20.2.4 weigand_init()	52
4.20.2.5 weigand_is_parity_ok()	53
Index	55

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

acs_msg_data_auth_req_t	5
acs_msg_data_auth_resp_t	5
acs_msg_data_door_ctrl_t	5
acs_msg_head_t	6
cache_item_t	6
reader_conf_t	7
reader_wiring_t	7
weigand26_buff_item_t	7
weigand26_frame_t	8
weigand26_t	8

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

acs_can_protocol.h	ACS CAN protocol	9
brownout.c	Brown-out detection	10
brownout.h	Brown-out detection	11
can_term_driver.c	Interface for LPC11C24's on-board CCAN driver in ROM	12
can_term_driver.h	Interface for LPC11C24's on-board CCAN driver in ROM	16
reader.c	RFID reader driver	19
reader.h	RFID card reader driver	22
start.c	Main entry point	24
static_cache.c	Statically allocated cache for user IDs	26
static_cache.h	Statically allocated cache for user IDs	26
storage.c	Storage implementation	29
storage.h	Storage implementation	31
terminal.c	Terminal client for access control system (ACS)	34
terminal.h	Terminal client for access control system (ACS)	37
terminal_config.c	Configuration of hardware and software	39
terminal_config.h	Terminal client configuration (hardware and software).	
•		

watchdog.c	
HW watchdog	44
watchdog.h	
HW watchdog	45
weigand.c	
Wiegand26 interface driver	47
weigand.h	
Wiegand26 interface driver	50

Chapter 3

Data Structure Documentation

3.1 acs_msg_data_auth_req_t Struct Reference

Data Fields

- `uint32_t user_id`

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

- [acs_can_protocol.h](#)

3.2 acs_msg_data_auth_resp_t Struct Reference

Data Fields

- `uint32_t user_id`

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

- [acs_can_protocol.h](#)

3.3 acs_msg_data_door_ctrl_t Struct Reference

Data Fields

- `uint8_t ctrl_command`

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

- [acs_can_protocol.h](#)

3.4 acs_msg_head_t Union Reference

Data Fields

-
- ```
struct {
 uint32_t src: ACS_ADDR_BITS
 uint32_t dst: ACS_ADDR_BITS
 uint32_t fc: ACS_FC_BITS
 uint32_t prio: ACS_PRIO_BITS
 uint32_t flags: 3
};
```
- **uint32\_t scalar**

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

- [acs\\_can\\_protocol.h](#)

## 3.5 cache\_item\_t Union Reference

```
#include <static_cache.h>
```

### Data Fields

- 
- ```
struct {
    uint32_t value: 2
    uint32_t key: 30
};
```
- **uint32_t scalar**

3.5.1 Detailed Description

Public types/enumerations/variables

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

- [static_cache.h](#)

3.6 reader_conf_t Struct Reference

Data Fields

- TimerHandle_t **timer_open**
- TimerHandle_t **timer_ok**
- uint16_t **open_time_sec**
- uint16_t **gled_time_sec**
- uint8_t **enabled**
- uint8_t **door_open**

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

- [reader.h](#)

3.7 reader_wiring_t Struct Reference

Data Fields

- uint8_t **data_port**
- uint8_t **d0_pin**
- uint8_t **d1_pin**
- uint8_t **beep_port**
- uint8_t **beep_pin**
- uint8_t **gled_port**
- uint8_t **gled_pin**
- uint8_t **rled_port**
- uint8_t **rled_pin**
- uint8_t **relay_port**
- uint8_t **relay_pin**
- uint8_t **sensor_port**
- uint8_t **sensor_pin**

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

- [reader.h](#)

3.8 weigand26_buff_item_t Struct Reference

Data Fields

- uint8_t **source**
- [weigand26_frame_t](#) **frame**

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

- [weigand.h](#)

3.9 weigand26_frame_t Union Reference

Data Fields

- ```
struct {
 uint32_t odd_parity: 1
 uint32_t card_number: 16
 uint32_t facility_code: 8
 uint32_t even_parity: 1
 uint32_t __pad0__: 6
};
```
- `uint32_t value`

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

- [weigand.h](#)

## 3.10 weigand26\_t Struct Reference

### Data Fields

- [weigand26\\_frame\\_t frame\\_buffer](#)
- `StreamBufferHandle_t consumer_buffer`
- `TimerHandle_t timer`
- `uint8_t frame_buffer_ptr`
- `uint8_t port`
- `uint8_t pin_d0`
- `uint8_t pin_d1`
- `uint8_t id`

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

- [weigand.c](#)

# Chapter 4

## File Documentation

### 4.1 acs\_can\_protocol.h File Reference

ACS CAN protocol.

#### Data Structures

- union `acs_msg_head_t`
- struct `acs_msg_data_auth_req_t`
- struct `acs_msg_data_auth_resp_t`
- struct `acs_msg_data_door_ctrl_t`

#### Macros

- #define `ACS_MSGOBJ_SEND_DOOR_A` 0
- #define `ACS_MSGOBJ_SEND_DOOR_B` 1
- #define `ACS_MSGOBJ_RECV_DOOR_A` 2
- #define `ACS_MSGOBJ_RECV_DOOR_B` 3
- #define `ACS_MSGOBJ_RECV_BCAST` 4
- #define `ACS_PRIO_BITS` 3
- #define `ACS_FC_BITS` 6
- #define `ACS_ADDR_BITS` 10
- #define `ACS_BROADCAST_ADDR` ((1 << ACS\_ADDR\_BITS) - 1)
- #define `ACS_RESERVED_ADDR` 0
- #define `ACS_MSTR_FIRST_ADDR` 1
- #define `ACS_MSTR_LAST_ADDR` 3
- #define `ACS_PNL_FIRST_ADDR` 4
- #define `ACS_PNL_LAST_ADDR` (ACS\_BROADCAST\_ADDR - 1)
- #define `ACS_SRC_ADDR_OFFSET` 0
- #define `ACS_DST_ADDR_OFFSET` (ACS\_SRC\_ADDR\_OFFSET + ACS\_ADDR\_BITS)
- #define `ACS_FC_OFFSET` (ACS\_DST\_ADDR\_OFFSET + ACS\_ADDR\_BITS)
- #define `ACS_PRIO_OFFSET` (ACS\_FC\_OFFSET + ACS\_FC\_BITS)
- #define `ACS_SRC_ADDR_MASK` (((1 << ACS\_ADDR\_BITS) - 1) << ACS\_SRC\_ADDR\_OFFSET)
- #define `ACS_DST_ADDR_MASK` (((1 << ACS\_ADDR\_BITS) - 1) << ACS\_DST\_ADDR\_OFFSET)
- #define `ACS_FC_MASK` (((1 << ACS\_FC\_BITS) - 1) << ACS\_FC\_OFFSET)
- #define `ACS_PRIO_MASK` (((1 << ACS\_PRIO\_BITS) - 1) << ACS\_PRIO\_OFFSET)

- #define **FC\_RESERVED** 0x0
- #define **FC\_USER\_AUTH\_REQ** 0x1
- #define **FC\_USER\_NOT\_AUTH\_RESP** 0x2
- #define **FC\_USER\_AUTH\_RESP** 0x3
- #define **FC\_DOOR\_CTRL** 0x4
- #define **FC\_DOOR\_STATUS** 0x5
- #define **FC\_ALIVE** 0x6
- #define **ACS\_MAX\_PRIO** 0
- #define **ACS\_LOW\_PRIO** ((1 << ACS\_PRIO\_BITS) - 1)
- #define **PRIOR\_RESERVED** 0x0
- #define **PRIOR\_USER\_AUTH\_REQ** 0x2
- #define **PRIOR\_USER\_AUTH\_RESP\_FAIL** 0x2
- #define **PRIOR\_USER\_AUTH\_RESP\_OK** 0x2
- #define **PRIOR\_DOOR\_CTRL** 0x3
- #define **PRIOR\_DOOR\_STATUS** 0x4
- #define **PRIOR\_ALIVE** 0x1
- #define **DATA\_DOOR\_CTRL\_REMOTE\_UNLCK** 0x01
- #define **DATA\_DOOR\_CTRL\_CLR\_CACHE** 0x02
- #define **DATA\_DOOR\_STATUS\_CLOSED** 0x01
- #define **DATA\_DOOR\_STATUS\_OPEN** 0x02
- #define **ACS\_MASTER\_ALIVE\_PERIOD\_MS** 5000
- #define **ACS\_MASTER\_ALIVE\_TIMEOUT\_MS** 10000

#### 4.1.1 Detailed Description

ACS CAN protocol.

Contains defines and data types for the protocol. ACS protocol uses messages with extended ID from CAN 2.0B specification.

Author

Petr Elexa

See also

LICENSE

## 4.2 brownout.c File Reference

Brown-out detection.

```
#include "brownout.h"
#include "board.h"
```

### Functions

- void **BOD\_IRQHandler** (void)
   
*Brown-out detection interrupt handler.*
- void **BOD\_Init** (void)
   
*Initialize brown-out detection.*

### 4.2.1 Detailed Description

Brown-out detection.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.2.2 Function Documentation

#### 4.2.2.1 BOD\_Init()

```
void BOD_Init (
 void)
```

Initialize brown-out detection.

#### 4.2.2.2 BOD\_IRQHandler()

```
void BOD_IRQHandler (
 void)
```

Brown-out detection interrupt handler.

## 4.3 brownout.h File Reference

Brown-out detection.

### Functions

- void [BOD\\_IRQHandler](#) (void)  
*Brown-out detection interrupt handler.*
- void [BOD\\_Init](#) (void)  
*Initialize brown-out detection.*

### 4.3.1 Detailed Description

Brown-out detection.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.3.2 Function Documentation

#### 4.3.2.1 BOD\_Init()

```
void BOD_Init (
 void)
```

Initialize brown-out detection.

#### 4.3.2.2 BOD\_IRQHandler()

```
void BOD_IRQHandler (
 void)
```

Brown-out detection interrupt handler.

## 4.4 can\_term\_driver.c File Reference

Interface for LPC11C24's on-board CCAN driver in ROM.

```
#include "can/can_term_driver.h"
#include <string.h>
```

### Macros

- #define **CCAN\_BCR\_QUANTA**(x) (((x) & 0x3F)
- #define **CCAN\_BCR\_SJW**(x) (((x) & 0x3) << 6)
- #define **CCAN\_BCR\_TSEG1**(x) (((x) & 0x0F) << 8)
- #define **CCAN\_BCR\_TSEG2**(x) (((x) & 0x07) << 12)
- #define **CAN\_CALC\_SYNC\_SEG** 1
- #define **TSEG1\_MIN** 1
- #define **TSEG1\_MAX** 13
- #define **TSEG2\_MIN** 1
- #define **TSEG2\_MAX** 8
- #define **SJW\_MAX** 4
- #define **BRP\_MIN** 1
- #define **BRP\_MAX** 32

## Functions

- static void `_timing_calculate` (uint32\_t baud\_rate, uint32\_t \*can\_api\_timing\_cfg)
- static void `_timing_calculate_sp` (uint32\_t baud\_rate, uint32\_t \*can\_api\_timing\_cfg)
- static void `_125_kbaud_75sp` (uint32\_t \*can\_api\_timing\_cfg)
- void `CAN_init` (CCAN\_CALLBACKS\_T \*ptr\_callbacks, uint32\_t baud\_rate)  
*Initializes CAN periphery.*
- void `CAN_recv_filter` (uint8\_t msgobj\_num, uint32\_t id, uint32\_t mask, bool extended)  
*Setup HW filter for received CAN messages.*
- void `CAN_recv_filter_all_ext` (uint8\_t msgobj\_num)  
*Setup HW filter to receive all extended frames ID (0-0x1FFFFFFF).*
- void `CAN_send_once` (uint8\_t msgobj\_num, uint32\_t id, uint8\_t \*data, uint8\_t size)  
*Send one time CAN message.*
- void `CAN_send_test` (void)  
*Send test message on CAN.*
- void `CAN_IRQHandler` (void)  
*CAN interrupt handler.*

### 4.4.1 Detailed Description

Interface for LPC11C24's on-board CCAN driver in ROM.

Based on LPCOpen CCAN on-chip driver example.

#### Author

Petr Elexa

#### Note

Copyright(C) NXP Semiconductors, 2012 All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the LPC products. This software is supplied "AS IS" without any warranties of any kind, and NXP Semiconductors and its licensor disclaim any and all warranties, express or implied, including all implied warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights. NXP Semiconductors assumes no responsibility or liability for the use of the software, conveys no license or rights under any patent, copyright, mask work right, or any other intellectual property rights in or to any products. NXP Semiconductors reserves the right to make changes in the software without notification. NXP Semiconductors also makes no representation or warranty that such application will be suitable for the specified use without further testing or modification.

Permission to use, copy, modify, and distribute this software and its documentation is hereby granted, under NXP Semiconductors' and its licensor's relevant copyrights in the software, without fee, provided that it is used in conjunction with NXP Semiconductors microcontrollers. This copyright, permission, and disclaimer notice must appear in all copies of this code.

## 4.4.2 Function Documentation

### 4.4.2.1 \_timing\_calculate()

```
static void _timing_calculate (
 uint32_t baud_rate,
 uint32_t * can_api_timing_cfg) [static]
```

Private functions

### 4.4.2.2 CAN\_init()

```
void CAN_init (
 CCAN_CALLBACKS_T * ptr_callbacks,
 uint32_t baud_rate)
```

Initializes CAN periphery.

Public functions

### 4.4.2.3 CAN\_IRQHandler()

```
void CAN_IRQHandler (
 void)
```

CAN interrupt handler.

The CAN interrupt handler must be provided by the user application.  
It's function is to call the handler located in the ROM.

### 4.4.2.4 CAN\_recv\_filter()

```
void CAN_recv_filter (
 uint8_t msgobj_num,
 uint32_t id,
 uint32_t mask,
 bool extended)
```

Setup HW filter for received CAN messages.

The filter matches when <recieved\_id> & mask == id & mask.  
Non-matching messages are dropped.

**Parameters**

|                   |                                               |
|-------------------|-----------------------------------------------|
| <i>msgobj_num</i> | ... number of message object (0-31) to setup. |
| <i>id</i>         | ... CAN message ID                            |
| <i>mask</i>       | ... CAN ID mask                               |
| <i>extended</i>   | ... use extended frame ID (29bit) if true     |

**4.4.2.5 CAN\_recv\_filter\_all\_ext()**

```
void CAN_recv_filter_all_ext (
 uint8_t msgobj_num)
```

Setup HW filter to receive all extended frames ID (0-0x1FFFFFFF).

This setups HW filter for received CAN message.  
The filter matches when <recieved\_id> & mask == id & mask.  
Non-matching messages are dropped.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| <i>msgobj_num</i> | ... number of message object (0-31). |
|-------------------|--------------------------------------|

**4.4.2.6 CAN\_send\_once()**

```
void CAN_send_once (
 uint8_t msgobj_num,
 uint32_t id,
 uint8_t * data,
 uint8_t size)
```

Send one time CAN message.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| <i>msgobj_num</i> | ... number of message object (0-31). |
| <i>id</i>         | ... CAN arbitration ID.              |
| <i>data</i>       | ... pointer to data to send.         |
| <i>size</i>       | ... size of data.                    |

**4.4.2.7 CAN\_send\_test()**

```
void CAN_send_test (
```

```
 void)
```

Send test message on CAN.

## 4.5 can\_term\_driver.h File Reference

Interface for LPC11C24's on-board CCAN driver in ROM.

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

### Macros

- #define **CCAN\_MSG\_OBJ\_FIRST** 0
- #define **CCAN\_MSG\_OBJ\_LAST** 31
- #define **CAN\_EXT\_ID\_BIT\_MASK** 0x1FFFFFFFUL
- #define **CAN\_DLC\_MAX** 8

### Functions

- void **CAN\_init** (CCAN\_CALLBACKS\_T \*ptr\_callbacks, uint32\_t baud\_rate)  
*Initializes CAN periphery.*
- void **CAN\_recv\_filter** (uint8\_t msgobj\_num, uint32\_t id, uint32\_t mask, bool extended)  
*Setup HW filter for received CAN messages.*
- void **CAN\_recv\_filter\_all\_ext** (uint8\_t msgobj\_num)  
*Setup HW filter to receive all extended frames ID (0-0x1FFFFFFF).*
- void **CAN\_send\_once** (uint8\_t msgobj\_num, uint32\_t id, uint8\_t \*data, uint8\_t size)  
*Send one time CAN message.*
- void **CAN\_send\_test** (void)  
*Send test message on CAN.*
- void **CAN\_IRQHandler** (void)  
*CAN interrupt handler.*

### 4.5.1 Detailed Description

Interface for LPC11C24's on-board CCAN driver in ROM.

#### Author

Petr Elexa

#### See also

LICENSE

## 4.5.2 Function Documentation

### 4.5.2.1 CAN\_init()

```
void CAN_init (
 CCAN_CALLBACKS_T * ptr_callbacks,
 uint32_t baud_rate)
```

Initializes CAN periphery.

Function should be executed before using the CAN bus.  
Initializes the CAN controller, on-chip drivers.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <i>ptr_callbacks</i> | ... pointer to callback structure |
| <i>baud_rate</i>     | ... CAN baud rate to use          |

#### Public functions

### 4.5.2.2 CAN\_IRQHandler()

```
void CAN_IRQHandler (
 void)
```

CAN interrupt handler.

The CAN interrupt handler must be provided by the user application.  
It's function is to call the handler located in the ROM.

### 4.5.2.3 CAN\_recv\_filter()

```
void CAN_recv_filter (
 uint8_t msgobj_num,
 uint32_t id,
 uint32_t mask,
 bool extended)
```

Setup HW filter for received CAN messages.

The filter matches when <recieved\_id> & mask == id & mask.  
Non-matching messages are dropped.

**Parameters**

|                   |                                               |
|-------------------|-----------------------------------------------|
| <i>msgobj_num</i> | ... number of message object (0-31) to setup. |
| <i>id</i>         | ... CAN message ID                            |
| <i>mask</i>       | ... CAN ID mask                               |
| <i>extended</i>   | ... use extended frame ID (29bit) if true     |

**4.5.2.4 CAN\_recv\_filter\_all\_ext()**

```
void CAN_recv_filter_all_ext (
 uint8_t msgobj_num)
```

Setup HW filter to receive all extended frames ID (0-0xFFFFFFFF).

This setups HW filter for received CAN message.  
The filter matches when <recieved\_id> & mask == id & mask.  
Non-matching messages are dropped.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| <i>msgobj_num</i> | ... number of message object (0-31). |
|-------------------|--------------------------------------|

**4.5.2.5 CAN\_send\_once()**

```
void CAN_send_once (
 uint8_t msgobj_num,
 uint32_t id,
 uint8_t * data,
 uint8_t size)
```

Send one time CAN message.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| <i>msgobj_num</i> | ... number of message object (0-31). |
| <i>id</i>         | ... CAN arbitration ID.              |
| <i>data</i>       | ... pointer to data to send.         |
| <i>size</i>       | ... size of data.                    |

**4.5.2.6 CAN\_send\_test()**

```
void CAN_send_test (
```

```
 void)
```

Send test message on CAN.

## 4.6 reader.c File Reference

RFID reader driver.

```
#include <reader.h>
```

### Macros

- #define **DOOR\_SENSOR\_VALUE\_OPEN** LOG\_LOW
- #define **DOOR\_OPEN** 1
- #define **DOOR\_CLOSED** 0

### Functions

- static void **\_timer\_open\_callback** (TimerHandle\_t pxTimer)
- static void **\_timer\_ok\_callback** (TimerHandle\_t pxTimer)
- void **reader\_init** (uint8\_t idx)  
*Initialize reader driver.*
- void **reader\_deinit** (uint8\_t idx)  
*Disable reader driver.*
- int8\_t **reader\_get\_request\_from\_buffer** (uint32\_t \*user\_id, uint16\_t time\_to\_wait\_ms)  
*Disable reader driver.*
- void **reader\_unlock** (uint8\_t idx, bool with\_beep, bool with\_ok\_led)  
*Unlock door belonging to reader.*
- bool **reader\_is\_door\_open** (uint8\_t reader\_idx)  
*Check door status.*
- void **reader\_sensor\_int\_handler** (uint8\_t port, uint32\_t int\_states)
- void **PIOINT0\_IRQHandler** (void)
- void **PIOINT1\_IRQHandler** (void)

### Variables

- static StreamBufferHandle\_t **\_reader\_buffer**
- static const **reader\_wiring\_t \_reader\_wiring** [ACS\_READER\_MAXCOUNT]
- **reader\_conf\_t reader\_conf** [ACS\_READER\_MAXCOUNT]  
*Configuration of readers.*

### 4.6.1 Detailed Description

RFID reader driver.

This RFID reader implementation uses Weigand26 card reader protocol.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.6.2 Function Documentation

#### 4.6.2.1 reader\_deinit()

```
void reader_deinit (
 uint8_t idx)
```

Disable reader driver.

##### Parameters

|            |                  |
|------------|------------------|
| <i>idx</i> | ... reader index |
|------------|------------------|

#### 4.6.2.2 reader\_get\_request\_from\_buffer()

```
int8_t reader_get_request_from_buffer (
 uint32_t * user_id,
 uint16_t time_to_wait_ms)
```

Disable reader driver.

##### Parameters

|                |                                              |
|----------------|----------------------------------------------|
| <i>user_id</i> | ... 24-bit number                            |
| <i>idx</i>     | ... reader index                             |
| ...            | time_to_wait_ms ... time to wait for request |

#### 4.6.2.3 reader\_init()

```
void reader_init (
 uint8_t idx)
```

Initialize reader driver.

#### 4.6.2.4 reader\_is\_door\_open()

```
bool reader_is_door_open (
 uint8_t reader_idx)
```

Check door status.

##### Parameters

|            |                  |
|------------|------------------|
| <i>idx</i> | ... reader index |
|------------|------------------|

##### Returns

true if door is open

#### 4.6.2.5 reader\_unlock()

```
void reader_unlock (
 uint8_t idx,
 bool with_beep,
 bool with_ok_led)
```

Unlock door belonging to reader.

##### Parameters

|                    |                            |
|--------------------|----------------------------|
| <i>idx</i>         | ... reader index           |
| <i>with_beep</i>   | ... true for sound signal  |
| <i>with_ok_led</i> | ... true for visual signal |

## 4.6.3 Variable Documentation

#### 4.6.3.1 reader\_conf

`reader_conf_t reader_conf[ACS_READER_MAXCOUNT]`

---

**Initial value:**

```
=
{
 {
 .timer_ok = NULL,
 .timer_open = NULL,
 .open_time_sec = ACS_READER_A_OPEN_TIME_MS,
 .gled_time_sec = ACS_READER_A_OK_GLED_TIME_MS,
 .enabled = ACS_READER_A_ENABLED,
 .door_open = DOOR_CLOSED
 },
 {
 .timer_ok = NULL,
 .timer_open = NULL,
 .open_time_sec = ACS_READER_B_OPEN_TIME_MS,
 .gled_time_sec = ACS_READER_B_OK_GLED_TIME_MS,
 .enabled = ACS_READER_B_ENABLED,
 .door_open = DOOR_CLOSED
 }
}
```

Configuration of readers.

## 4.7 reader.h File Reference

RFID card reader driver.

```
#include <stdint.h>
#include <stdbool.h>
#include "board.h"
#include "FreeRTOS.h"
#include "stream_buffer.h"
#include "timers.h"
#include "weigand.h"
```

### Data Structures

- struct [reader\\_conf\\_t](#)
- struct [reader\\_wiring\\_t](#)

### Enumerations

- enum [reader\\_mode\\_t](#) { **READER\_MODE\_DEF** = 0, **READER\_MODE\_LOCKED**, **READER\_MODE\_UNLOCKED**, **ARN** }

### Functions

- void [reader\\_init](#) (uint8\_t idx)
 

*Initialize reader driver.*
- void [reader\\_deinit](#) (uint8\_t idx)
 

*Disable reader driver.*
- int8\_t [reader\\_get\\_request\\_from\\_buffer](#) (uint32\_t \*user\_id, uint16\_t time\_to\_wait\_ms)
 

*Disable reader driver.*
- void [reader\\_unlock](#) (uint8\_t idx, bool with\_beep, bool with\_ok\_led)
 

*Unlock door belonging to reader.*
- bool [reader\\_is\\_door\\_open](#) (uint8\_t reader\_idx)
 

*Check door status.*

## Variables

- `reader_conf_t reader_conf [ACS_READER_COUNT]`  
*Configuration of readers.*

### 4.7.1 Detailed Description

RFID card reader driver.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.7.2 Function Documentation

#### 4.7.2.1 reader\_deinit()

```
void reader_deinit (
 uint8_t idx)
```

Disable reader driver.

#### Parameters

|                  |                  |
|------------------|------------------|
| <code>idx</code> | ... reader index |
|------------------|------------------|

#### 4.7.2.2 reader\_get\_request\_from\_buffer()

```
int8_t reader_get_request_from_buffer (
 uint32_t * user_id,
 uint16_t time_to_wait_ms)
```

Disable reader driver.

#### Parameters

|                      |                                              |
|----------------------|----------------------------------------------|
| <code>user_id</code> | ... 24-bit number                            |
| <code>idx</code>     | ... reader index                             |
| ...                  | time_to_wait_ms ... time to wait for request |

#### 4.7.2.3 reader\_init()

```
void reader_init (
 uint8_t idx)
```

Initialize reader driver.

#### 4.7.2.4 reader\_is\_door\_open()

```
bool reader_is_door_open (
 uint8_t reader_idx)
```

Check door status.

##### Parameters

|            |                  |
|------------|------------------|
| <i>idx</i> | ... reader index |
|------------|------------------|

##### Returns

true if door is open

#### 4.7.2.5 reader\_unlock()

```
void reader_unlock (
 uint8_t idx,
 bool with_beep,
 bool with_ok_led)
```

Unlock door belonging to reader.

##### Parameters

|                    |                            |
|--------------------|----------------------------|
| <i>idx</i>         | ... reader index           |
| <i>with_beep</i>   | ... true for sound signal  |
| <i>with_ok_led</i> | ... true for visual signal |

## 4.8 start.c File Reference

Main entry point.

```
#include "can/can_term_driver.h"
#include "terminal.h"
#include "board.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"
#include "watchdog.h"
#include "brownout.h"
#include "storage.h"
```

## Functions

- static void **\_check\_system\_stack\_size** (void)
- int **main** (void)
- void **vConfigureTimerForRunTimeStats** (void)
- void **vApplicationMallocFailedHook** (void)
- void **vApplicationIdleHook** (void)
- void **vApplicationStackOverflowHook** (TaskHandle\_t pxTask, char \*pcTaskName)
- void **vApplicationTickHook** (void)

### 4.8.1 Detailed Description

Main entry point.

Contains start-up sequence.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.8.2 Function Documentation

#### 4.8.2.1 **\_check\_system\_stack\_size()**

```
static void _check_system_stack_size (
 void) [static]
```

Private types/enumerations/variables Public types/enumerations/variables Private functions

#### 4.8.2.2 main()

```
int main (
 void)
```

Public functions

## 4.9 static\_cache.c File Reference

Statically allocated cache for user IDs.

```
#include "static_cache.h"
#include <string.h>
```

### 4.9.1 Detailed Description

Statically allocated cache for user IDs.

It is similar to Set Associative Cache. Place key and value into sets by key hash. Each set is a sorted array.

#### Author

Petr Elexa

#### See also

LICENSE

## 4.10 static\_cache.h File Reference

Statically allocated cache for user IDs.

```
#include <stdint.h>
#include <stdbool.h>
#include "terminal_config.h"
```

## Data Structures

- union [cache\\_item\\_t](#)

## Macros

- #define STATIC\_CACHE\_SETS 4
- #define STATIC\_CACHE\_SET\_CAP 128
- #define STATIC\_CACHE\_CAPACITY (STATIC\_CACHE\_SET\_CAP \* STATIC\_CACHE\_SETS)

## Functions

- `bool static_cache_get (cache_item_t *ptr_kv)`  
*Retrieve item from the cache.*
- `void static_cache_insert (const cache_item_t kv)`  
*Insert item to the cache.*
- `void static_cache_erase (const cache_item_t kv)`  
*Erase item from the cache.*
- `void static_cache_reset (void)`  
*Clear all data in the cache.*
- `cache_item_t static_cache_convert (uint32_t key, uint32_t value)`  
*Create cache item from key and value parameters.*

### 4.10.1 Detailed Description

Statically allocated cache for user IDs.

It is similar to Set Associative Cache. Divide key and value into sets by key hash. Each set is a sorted array.

The key in cache can be up to 30 bits in size.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.10.2 Macro Definition Documentation

#### 4.10.2.1 STATIC\_CACHE\_SETS

```
#define STATIC_CACHE_SETS 4
```

Configuration of the static cache.

### 4.10.3 Function Documentation

#### 4.10.3.1 static\_cache\_convert()

```
cache_item_t static_cache_convert (
 uint32_t key,
 uint32_t value)
```

Create cache item from key and value parameters.

**Parameters**

|              |                           |
|--------------|---------------------------|
| <i>key</i>   | ... Key for the value.    |
| <i>value</i> | ... Value for key to use. |

**Returns**

Cache item structure.

**4.10.3.2 static\_cache\_erase()**

```
void static_cache_erase (
 const cache_item_t kv)
```

Erase item from the cache.

Complexity is  $O(\log(\text{STATIC\_CACHE\_SET\_CAP}) + 2 * (\text{STATIC\_CACHE\_SET\_CAP}))$ .

**Parameters**

|           |                                                                               |
|-----------|-------------------------------------------------------------------------------|
| <i>kv</i> | ... Key containing key to be erased. The item will be erased if key is found. |
|-----------|-------------------------------------------------------------------------------|

**4.10.3.3 static\_cache\_get()**

```
bool static_cache_get (
 cache_item_t * ptr_kv)
```

Retrieve item from the cache.

**Public functions**

Complexity is  $O(\log(\text{STATIC\_CACHE\_SET\_CAP}))$ .

**Parameters**

|               |                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------|
| <i>ptr_kv</i> | ... Pointer to <code>cache_item_t</code> containing key. The value will be filled in if key is found. |
|---------------|-------------------------------------------------------------------------------------------------------|

**Returns**

true if key is found.

#### 4.10.3.4 static\_cache\_insert()

```
void static_cache_insert (
 const cache_item_t kv)
```

Insert item to the cache.

Will overwrite items already in cache if the cache is full. Will also update item with same key.

Complexity is  $O(\log(\text{STATIC\_CACHE\_SET\_CAP}) + 2 * (\text{STATIC\_CACHE\_SET\_CAP}))$ .

##### Parameters

|           |                                   |
|-----------|-----------------------------------|
| <i>kv</i> | ... Key and value to be inserted. |
|-----------|-----------------------------------|

#### 4.10.3.5 static\_cache\_reset()

```
void static_cache_reset (
 void)
```

Clear all data in the cache.

Resets all data in cache to 0s.  
Complexity is  $O(\text{STATIC\_CACHE\_CAPACITY})$ .

## 4.11 storage.c File Reference

Storage implementation.

```
#include "storage.h"
#include "FreeRTOS.h"
#include "task.h"
```

## Functions

- static void **\_init\_I2C\_pins** (void)
- static void **\_wait\_for\_dev\_ready** (void)
- void **storage\_init** (void)
 

*Initialize I2C bus for storage.*
- bool **storage\_read\_word\_le** (const uint8\_t addr, uint16\_t \*data)
 

*Read a word from address (little-endian).*
- bool **storage\_write\_word\_le** (const uint8\_t addr, const uint16\_t data)
 

*Write a word to address (little-endian).*
- bool **storage\_read\_byte** (const uint8\_t addr, uint8\_t \*data)
 

*Read a byte from address.*
- bool **storage\_write\_byte** (const uint8\_t addr, const uint8\_t data)
 

*Write a byte to address.*
- void **I2C\_IRQHandler** (void)
 

*I2C Interrupt Handler.*

### 4.11.1 Detailed Description

Storage implementation.

Supports I2C EEPROMs and I/O expanders.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.11.2 Function Documentation

#### 4.11.2.1 I2C\_IRQHandler()

```
void I2C_IRQHandler (
 void)
```

I2C Interrupt Handler.

#### Returns

None

#### 4.11.2.2 storage\_init()

```
void storage_init (
 void)
```

Initialize I2C bus for storage.

#### 4.11.2.3 storage\_read\_byte()

```
bool storage_read_byte (
 const uint8_t addr,
 uint8_t * data)
```

Read a byte from address.

#### Returns

true if succeeded

#### 4.11.2.4 storage\_read\_word\_le()

```
bool storage_read_word_le (
 const uint8_t addr,
 uint16_t * data)
```

Read a word from address (little-endian).

##### Returns

true if succeeded

#### 4.11.2.5 storage\_write\_byte()

```
bool storage_write_byte (
 const uint8_t addr,
 const uint8_t data)
```

Write a byte to address.

##### Returns

true if succeeded

#### 4.11.2.6 storage\_write\_word\_le()

```
bool storage_write_word_le (
 const uint8_t addr,
 const uint16_t data)
```

Write a word to address (little-endian).

##### Returns

true if succeeded

## 4.12 storage.h File Reference

Storage implementation.

```
#include "board.h"
#include <stdint.h>
#include <stdbool.h>
```

## Macros

- #define **STORE\_I2C\_DEV** I2C0

## Functions

- void **storage\_init** (void)  
*Initialize I2C bus for storage.*
- bool **storage\_read\_word\_le** (const uint8\_t addr, uint16\_t \*data)  
*Read a word from address (little-endian).*
- bool **storage\_write\_word\_le** (const uint8\_t addr, const uint16\_t data)  
*Write a word to address (little-endian).*
- bool **storage\_read\_byte** (const uint8\_t addr, uint8\_t \*data)  
*Read a byte from address.*
- bool **storage\_write\_byte** (const uint8\_t addr, const uint8\_t data)  
*Write a byte to address.*

### 4.12.1 Detailed Description

Storage implementation.

Supports I2C EEPROMs and I/O expanders.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.12.2 Function Documentation

#### 4.12.2.1 **storage\_init()**

```
void storage_init (
 void)
```

Initialize I2C bus for storage.

#### 4.12.2.2 storage\_read\_byte()

```
bool storage_read_byte (
 const uint8_t addr,
 uint8_t * data)
```

Read a byte from address.

##### Returns

true if succeeded

#### 4.12.2.3 storage\_read\_word\_le()

```
bool storage_read_word_le (
 const uint8_t addr,
 uint16_t * data)
```

Read a word from address (little-endian).

##### Returns

true if succeeded

#### 4.12.2.4 storage\_write\_byte()

```
bool storage_write_byte (
 const uint8_t addr,
 const uint8_t data)
```

Write a byte to address.

##### Returns

true if succeeded

#### 4.12.2.5 storage\_write\_word\_le()

```
bool storage_write_word_le (
 const uint8_t addr,
 const uint16_t data)
```

Write a word to address (little-endian).

##### Returns

true if succeeded

## 4.13 terminal.c File Reference

Terminal client for access control system (ACS).

```
#include "terminal.h"
#include "board.h"
#include "weigand.h"
#include "static_cache.h"
#include "FreeRTOS.h"
#include "task.h"
#include "stream_buffer.h"
#include "can/can_term_driver.h"
#include "acs_can_protocol.h"
#include <stdio.h>
#include <string.h>
```

### Typedefs

- `typedef cache_item_t term_cache_item_t`

### Enumerations

- `enum term_cache_reader { cache_reader_none = 0, cache_reader_A = 1, cache_reader_B = 2, cache_reader_all = 3 }`

### Functions

- `static uint8_t map_reader_idx_to_cache (uint8_t reader_idx)`
- `static void _terminal_userAuthorized (uint8_t reader_idx)`
- `static void __terminal_userNotAuthorized (uint8_t reader_idx)`
- `static void _timerCallback (TimerHandle_t pxTimer)`
- `void termCanError (uint32_t error_info)`  
*CAN error callback.*
- `void termCanSend (uint8_t msg_obj_num)`  
*CAN transmit callback.*
- `void termCanRecv (uint8_t msg_obj_num)`  
*CAN receive callback.*
- `static void terminalSendDoorStatus (uint8_t reader_idx, bool is_open)`
- `static void terminalRequestAuth (uint32_t user_id, uint8_t reader_idx)`
- `static void terminalUserIdentified (uint32_t user_id, uint8_t reader_idx)`
- `static void terminalTask (void *pvParameters)`
- `void terminalInit (void)`  
*Initialize terminal.*
- `void terminalReconfigure (reader_conf_t *reader_cfg, uint8_t reader_idx)`  
*Reconfigure terminal's reader at runtime.*

## Variables

- static const uint16\_t **USER\_REQUEST\_WAIT\_MS** = 1500
- static uint16\_t **\_act\_master** = ACS\_RESERVED\_ADDR
- static bool **\_master\_timeout** = true
- static TimerHandle\_t **\_act\_timer** = NULL
- static const uint32\_t **\_act\_timer\_id** = TERMINAL\_TIMER\_ID
- static bool **\_last\_door\_state** [ACS\_READER\_COUNT] = {false, false}

### 4.13.1 Detailed Description

Terminal client for access control system (ACS).

#### Author

Petr Elexa

#### See also

LICENSE

### 4.13.2 Function Documentation

#### 4.13.2.1 map\_reader\_idx\_to\_cache()

```
static uint8_t map_reader_idx_to_cache (
 uint8_t reader_idx) [inline], [static]
```

Private functions

#### 4.13.2.2 term\_can\_error()

```
void term_can_error (
 uint32_t error_info)
```

CAN error callback.

Public functions

#### 4.13.2.3 term\_can\_recv()

```
void term_can_recv (
 uint8_t msg_obj_num)
```

CAN receive callback.

Function is executed by the Callback handler after a CAN message has been received.

**Parameters**

|                    |                                                                                    |
|--------------------|------------------------------------------------------------------------------------|
| <i>msg_obj_num</i> | Contains the number of the message object that triggered the CAN receive callback. |
|--------------------|------------------------------------------------------------------------------------|

**4.13.2.4 term\_can\_send()**

```
void term_can_send (
 uint8_t msg_obj_num)
```

CAN transmit callback.

Function is executed by the Callback handler after a CAN message has been transmitted.

**Parameters**

|                    |                                                                                     |
|--------------------|-------------------------------------------------------------------------------------|
| <i>msg_obj_num</i> | Contains the number of the message object that triggered the CAN transmit callback. |
|--------------------|-------------------------------------------------------------------------------------|

**4.13.2.5 terminal\_init()**

```
void terminal_init (
 void)
```

Initialize terminal.

Will initialize terminal configuration and all readers and create terminal task.

**Parameters**

|                   |  |
|-------------------|--|
| <i>reader_cfg</i> |  |
| <i>id</i>         |  |

**4.13.2.6 terminal\_reconfigure()**

```
void terminal_reconfigure (
 reader_conf_t * reader_cfg,
 uint8_t id)
```

Reconfigure terminal's reader at runtime.

**Parameters**

|                   |  |
|-------------------|--|
| <i>reader_cfg</i> |  |
| <i>id</i>         |  |

### 4.13.3 Variable Documentation

#### 4.13.3.1 USER\_REQUEST\_WAIT\_MS

```
const uint16_t USER_REQUEST_WAIT_MS = 1500 [static]
```

Private types/enumerations/variables

## 4.14 terminal.h File Reference

Terminal client for access control system (ACS).

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

### Functions

- void `terminal_init` (void)  
*Initialize terminal.*
- void `terminal_reconfigure` (`reader_conf_t` \*`reader_cfg`, `uint8_t` `id`)  
*Reconfigure terminal's reader at runtime.*
- void `term_can_recv` (`uint8_t` `msg_obj_num`)  
*CAN receive callback.*
- void `term_can_send` (`uint8_t` `msg_obj_num`)  
*CAN transmit callback.*
- void `term_can_error` (`uint32_t` `error_info`)  
*CAN error callback.*

### 4.14.1 Detailed Description

Terminal client for access control system (ACS).

Author

Petr Elexa

See also

LICENSE

### 4.14.2 Function Documentation

#### 4.14.2.1 term\_can\_error()

```
void term_can_error (
 uint32_t error_info)
```

CAN error callback.

Function is executed by the Callback handler after an error has occurred on the CAN bus.

**Parameters**

|                   |                                                                |
|-------------------|----------------------------------------------------------------|
| <i>error_info</i> | Contains the error code that triggered the CAN error callback. |
|-------------------|----------------------------------------------------------------|

**Public functions****4.14.2.2 term\_can\_recv()**

```
void term_can_recv (
 uint8_t msg_obj_num)
```

CAN receive callback.

Function is executed by the Callback handler after a CAN message has been received.

**Parameters**

|                    |                                                                                    |
|--------------------|------------------------------------------------------------------------------------|
| <i>msg_obj_num</i> | Contains the number of the message object that triggered the CAN receive callback. |
|--------------------|------------------------------------------------------------------------------------|

**4.14.2.3 term\_can\_send()**

```
void term_can_send (
 uint8_t msg_obj_num)
```

CAN transmit callback.

Function is executed by the Callback handler after a CAN message has been transmitted.

**Parameters**

|                    |                                                                                     |
|--------------------|-------------------------------------------------------------------------------------|
| <i>msg_obj_num</i> | Contains the number of the message object that triggered the CAN transmit callback. |
|--------------------|-------------------------------------------------------------------------------------|

**4.14.2.4 terminal\_init()**

```
void terminal_init (
 void)
```

Initialize terminal.

Will initialize terminal configuration and all readers and create terminal task.

**Parameters**

|                   |  |
|-------------------|--|
| <i>reader_cfg</i> |  |
| <i>id</i>         |  |

#### 4.14.2.5 terminal\_reconfigure()

```
void terminal_reconfigure (
 reader_conf_t * reader_cfg,
 uint8_t id)
```

Reconfigure terminal's reader at runtime.

##### Parameters

|                   |  |
|-------------------|--|
| <i>reader_cfg</i> |  |
| <i>id</i>         |  |

## 4.15 terminal\_config.c File Reference

Configuration of hardware and software.

```
#include "terminal_config.h"
#include "storage.h"
#include "acs_can_protocol.h"
```

### Macros

- #define **ACS\_ADDR\_BIT\_MASK** ((1 << ACS\_ADDR\_BITS) - 1)

### Functions

- uint16\_t **get\_reader\_a\_addr** (void)
 

*Get network address of door A.*
- uint16\_t **get\_reader\_b\_addr** (void)
 

*Get network address of door B.*
- static bool **\_load\_acs\_addrs\_from\_ext\_stor** (void)
- static bool **\_save\_acs\_addrs\_from\_ext\_stor** (void)
- void **set\_reader\_addr** (uint16\_t acs\_addr)
 

*Address setter.*
- bool **terminal\_config\_init** (void)
 

*Initialize configuration for terminal.*

### Variables

- uint16\_t **\_READER\_A\_ADDR** = 0x4
- uint16\_t **\_READER\_B\_ADDR** = 0x5

### 4.15.1 Detailed Description

Configuration of hardware and software.

Author

Petr Elexa

See also

LICENSE

### 4.15.2 Function Documentation

#### 4.15.2.1 get\_reader\_a\_addr()

```
uint16_t get_reader_a_addr (
 void) [inline]
```

Get network address of door A.

#### 4.15.2.2 get\_reader\_b\_addr()

```
uint16_t get_reader_b_addr (
 void) [inline]
```

Get network address of door B.

#### 4.15.2.3 set\_reader\_addr()

```
void set_reader_addr (
 const uint16_t acs_addr)
```

Address setter.

Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>acs_addr</i> | ... ACS network address (odd or even). |
|-----------------|----------------------------------------|

#### 4.15.2.4 terminal\_config\_init()

```
bool terminal_config_init (
 void)
```

Initialize configuration for terminal.

Reads address from storage.

Returns

true if succeeded

## 4.16 terminal\_config.h File Reference

Terminal client configuration (hardware and software). \*.

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

### Macros

- #define **DEVEL\_BOARD**
- #define **ENABLE\_LOCAL\_ACS\_ADDR\_WRITE** 0
- #define **CACHING\_ENABLED** 0
- #define **CAN\_BAUD\_RATE** 125000
- #define **STORE\_I2C\_BUS\_FREQ** 400000
- #define **STORE\_I2C\_SLAVE\_ADDR** 0x50
- #define **ACS\_READER\_A\_IDX** 0
- #define **ACS\_READER\_B\_IDX** 1
- #define **ACS\_READER\_COUNT** 2
- #define **BEEP\_ON\_SUCCESS** false
- #define **OK\_LED\_ON\_SUCCESS** true
- #define **SENSOR\_IS\_NO** 0
- #define **SENSOR\_IS\_NC** 1
- #define **DOOR\_SENSOR\_TYPE** SENSOR\_IS\_NO
- #define **ACS\_COMM\_STATUS\_LED\_PORT** 0
- #define **ACS\_COMM\_STATUS\_LED\_PIN** 6
- #define **IAP\_READ\_UID** 58
- #define **PTR\_READER\_FIRST\_ADDR** 0x0
- #define **STORE\_DEV\_BUSY\_FOR** 50
- #define **ACS\_READER\_A\_ENABLED** true
- #define **ACS\_READER\_A\_DATA\_PORT** 3
- #define **ACS\_READER\_A\_D0\_PIN** 2
- #define **ACS\_READER\_A\_D1\_PIN** 1
- #define **ACS\_READER\_A\_BEEP\_PORT** 2
- #define **ACS\_READER\_A\_BEEP\_PIN** 7
- #define **ACS\_READER\_A\_GLED\_PORT** 2
- #define **ACS\_READER\_A\_GLED\_PIN** 1
- #define **ACS\_READER\_A\_RLED\_PORT** 2

- #define **ACS\_READER\_A\_RLED\_PIN** 0
- #define **ACS\_READER\_A\_RELAY\_PORT** 1
- #define **ACS\_READER\_A\_RELAY\_PIN** 10
- #define **ACS\_READER\_A\_SENSOR\_PORT** 1
- #define **ACS\_READER\_A\_SENSOR\_PIN** 8
- #define **ACS\_READER\_A\_OPEN\_TIME\_MS** 8000
- #define **ACS\_READER\_A\_OK\_GLED\_TIME\_MS** 4000
- #define **ACS\_READER\_B\_ENABLED** true
- #define **ACS\_READER\_B\_DATA\_PORT** 2
- #define **ACS\_READER\_B\_D0\_PIN** 8
- #define **ACS\_READER\_B\_D1\_PIN** 6
- #define **ACS\_READER\_B\_BEEP\_PORT** 2
- #define **ACS\_READER\_B\_BEEP\_PIN** 10
- #define **ACS\_READER\_B\_GLED\_PORT** 2
- #define **ACS\_READER\_B\_GLED\_PIN** 3
- #define **ACS\_READER\_B\_RLED\_PORT** 2
- #define **ACS\_READER\_B\_RLED\_PIN** 2
- #define **ACS\_READER\_B\_RELAY\_PORT** 1
- #define **ACS\_READER\_B\_RELAY\_PIN** 11
- #define **ACS\_READER\_B\_SENSOR\_PORT** 1
- #define **ACS\_READER\_B\_SENSOR\_PIN** 5
- #define **ACS\_READER\_B\_OPEN\_TIME\_MS** 8000
- #define **ACS\_READER\_B\_OK\_GLED\_TIME\_MS** 4000
- #define **WEIGAND\_DEVICE\_LIMIT** 4
- #define **SERIAL\_DEVICE\_LIMIT** 0
- #define **ACS\_READER\_MAXCOUNT** 2
- #define **TERMINAL\_TIMER\_ID** 15
- #define **HW\_WATCHDOG\_TIMEOUT** 1
- #define **LOG\_HIGH** 1
- #define **LOG\_LOW** 0

## Functions

- uint16\_t [get\\_reader\\_a\\_addr](#) (void)  
*Get network address of door A.*
- uint16\_t [get\\_reader\\_b\\_addr](#) (void)  
*Get network address of door B.*
- void [set\\_reader\\_addr](#) (const uint16\_t acs\_addr)  
*Address setter.*
- bool [terminal\\_config\\_init](#) (void)  
*Initialize configuration for terminal.*

## Variables

- unsigned int **TERMINAL\_UID** [5]

### 4.16.1 Detailed Description

Terminal client configuration (hardware and software). \*.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.16.2 Function Documentation

#### 4.16.2.1 get\_reader\_a\_addr()

```
uint16_t get_reader_a_addr (
 void) [inline]
```

Get network address of door A.

#### 4.16.2.2 get\_reader\_b\_addr()

```
uint16_t get_reader_b_addr (
 void) [inline]
```

Get network address of door B.

#### 4.16.2.3 set\_reader\_addr()

```
void set_reader_addr (
 const uint16_t acs_addr)
```

Address setter.

#### Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>acs_addr</i> | ... ACS network address (odd or even). |
|-----------------|----------------------------------------|

#### 4.16.2.4 terminal\_config\_init()

```
bool terminal_config_init (
 void)
```

Initialize configuration for terminal.

Reads address from storage.

##### Returns

true if succeeded

## 4.17 watchdog.c File Reference

HW watchdog.

```
#include "watchdog.h"
#include "board.h"
```

### Functions

- void [WDT\\_IRQHandler](#) (void)  
*Watchdog timer interrupt handler.*
- void [WDT\\_Init](#) (uint8\_t timeout)  
*Initialize Watchdog timer.*
- void [WDT\\_Feed](#) (void)  
*Feed watchdog timer.*

### 4.17.1 Detailed Description

HW watchdog.

#### Author

Petr Elexa

#### Note

Copyright(C) NXP Semiconductors, 2012 All rights reserved.

Software that is described herein is for illustrative purposes only which provides customers with programming information regarding the LPC products. This software is supplied "AS IS" without any warranties of any kind, and NXP Semiconductors and its licensor disclaim any and all warranties, express or implied, including all implied warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights. NXP Semiconductors assumes no responsibility or liability for the use of the software, conveys no license or rights under any patent, copyright, mask work right, or any other intellectual property rights in or to any products. NXP Semiconductors reserves the right to make changes in the software without notification. NXP Semiconductors also makes no representation or warranty that such application will be suitable for the specified use without further testing or modification.

Permission to use, copy, modify, and distribute this software and its documentation is hereby granted, under NXP Semiconductors' and its licensor's relevant copyrights in the software, without fee, provided that it is used in conjunction with NXP Semiconductors microcontrollers. This copyright, permission, and disclaimer notice must appear in all copies of this code.

## 4.17.2 Function Documentation

### 4.17.2.1 WDT\_Feed()

```
void WDT_Feed (
 void)
```

Feed watchdog timer.

### 4.17.2.2 WDT\_Init()

```
void WDT_Init (
 uint8_t timeout)
```

Initialize Watchdog timer.

After init watchdog must be reloaded by @ref WDT\_Feed.

#### Parameters

|                |                                                     |
|----------------|-----------------------------------------------------|
| <i>timeout</i> | ... time to reset or interrupt from watchdog timer. |
|----------------|-----------------------------------------------------|

### 4.17.2.3 WDT\_IRQHandler()

```
void WDT_IRQHandler (
 void)
```

Watchdog timer interrupt handler.

Public functions

## 4.18 watchdog.h File Reference

HW watchdog.

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

## Functions

- void [WDT\\_IRQHandler](#) (void)  
*Watchdog timer interrupt handler.*
- void [WDT\\_Init](#) (uint8\_t timeout)  
*Initialize Watchdog timer.*
- void [WDT\\_Feed](#) (void)  
*Feed watchdog timer.*

### 4.18.1 Detailed Description

HW watchdog.

#### Author

Petr Elexa

#### See also

LICENSE

### 4.18.2 Function Documentation

#### 4.18.2.1 WDT\_Feed()

```
void WDT_Feed (
 void)
```

Feed watchdog timer.

#### 4.18.2.2 WDT\_Init()

```
void WDT_Init (
 uint8_t timeout)
```

Initialize Watchdog timer.

After init watchdog must be reloaded by @ref WDT\_Feed.

#### Parameters

|                |                                                     |
|----------------|-----------------------------------------------------|
| <i>timeout</i> | ... time to reset or interrupt from watchdog timer. |
|----------------|-----------------------------------------------------|

### 4.18.2.3 WDT\_IRQHandler()

```
void WDT_IRQHandler (
 void)
```

Watchdog timer interrupt handler.

*Handles watchdog timer timeout events.  
Will be called only in debug mode.*

Public functions

## 4.19 weigand.c File Reference

Wiegand26 interface driver.

```
#include "weigand.h"
#include "board.h"
#include "timers.h"
#include <limits.h>
```

### Data Structures

- struct [weigand26\\_t](#)

### Functions

- static void [weigand\\_frame\\_timeout](#) (TimerHandle\_t pxTimer)
- void [weigand\\_init](#) (StreamBufferHandle\_t buffer, uint8\_t id, uint8\_t dx\_port, uint8\_t d0\_pin, uint8\_t d1\_pin)
 *Initialize Wiegand driver.*
- void [weigand\\_disable](#) (uint8\_t dx\_port, uint8\_t d0\_pin, uint8\_t d1\_pin)
 *Disable Wiegand driver.*
- bool [weigand\\_pending\\_frame](#) ([weigand26\\_t](#) \*device)
- [weigand26\\_frame\\_t weigand\\_get\\_frame](#) ([weigand26\\_t](#) \*device)
- uint32\_t [weigand\\_get\\_facility](#) ([weigand26\\_frame\\_t](#) frame)
 *Parse facility code from frame.*
- uint32\_t [weigand\\_get\\_card](#) ([weigand26\\_frame\\_t](#) frame)
 *Parse card number from frame.*
- bool [weigand\\_is\\_parity\\_ok](#) ([weigand26\\_frame\\_t](#) frame)
 *Check frame parity.*
- static void [\\_wake\\_timer\\_on\\_frame\\_start](#) ([weigand26\\_t](#) \*device)
- void [weigand\\_int\\_handler](#) ([weigand26\\_t](#) \*device)
- void [PIOINT2\\_IRQHandler](#) (void)
- void [PIOINT3\\_IRQHandler](#) (void)

## Variables

- static `weigand26_t device [WEIGAND_DEVICE_LIMIT] = {0}`

### 4.19.1 Detailed Description

Wiegand26 interface driver.

500 b/s transfer rate data pulse 40-70us data interval >2ms

Only Wiegand 26bit format Frame: | even parity (1b) | facility code (8b) | card number (16b) | odd parity (1b) | transmission duration ~52ms

#### Author

Petr Elexa

#### See also

LICENSE

### 4.19.2 Function Documentation

#### 4.19.2.1 weigand\_disable()

```
void weigand_disable (
 uint8_t dx_port,
 uint8_t d0_pin,
 uint8_t d1_pin)
```

Disable Wiegand driver.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>dx_port</code> | ... Port number for data signals. |
| <code>d0_pin</code>  | ... Pin for 0's data signal.      |
| <code>d1_pin</code>  | ... Pin for 1's data signal.      |

#### 4.19.2.2 weigand\_get\_card()

```
uint32_t weigand_get_card (
 weigand26_frame_t frame)
```

Parse card number from frame.

**Parameters**

|              |                           |
|--------------|---------------------------|
| <i>frame</i> | ... Wiegand26 data frame. |
|--------------|---------------------------|

**Returns**

card number

**4.19.2.3 weigand\_get\_facility()**

```
uint32_t weigand_get_facility (
 weigand26_frame_t frame)
```

Parse facility code from frame.

**Parameters**

|              |                           |
|--------------|---------------------------|
| <i>frame</i> | ... Wiegand26 data frame. |
|--------------|---------------------------|

**Returns**

facility code

**4.19.2.4 weigand\_init()**

```
void weigand_init (
 StreamBufferHandle_t buffer,
 uint8_t id,
 uint8_t dx_port,
 uint8_t d0_pin,
 uint8_t d1_pin)
```

Initialize Wiegand driver.

**Note**

One buffer for each consumer is preferred.

**Parameters**

|                |                                                                                              |
|----------------|----------------------------------------------------------------------------------------------|
| <i>buffer</i>  | ... Receive buffer for frames from RFID card/tags. See <a href="#">weigand26_buff_item_t</a> |
| <i>id</i>      | ... interface identification                                                                 |
| <i>dx_port</i> | ... Port number for data signals.                                                            |
| <i>d0_pin</i>  | ... Pin for 0's data signal.                                                                 |
| <i>d1_pin</i>  | ... Pin for 1's data signal.                                                                 |

#### 4.19.2.5 weigand\_is\_parity\_ok()

```
bool weigand_is_parity_ok (
 weigand26_frame_t frame)
```

Check frame parity.

##### Parameters

|              |                           |
|--------------|---------------------------|
| <i>frame</i> | ... Wiegand26 data frame. |
|--------------|---------------------------|

##### Returns

true if parity is valid

## 4.20 weigand.h File Reference

Wiegand26 interface driver.

```
#include <stdbool.h>
#include <stdint.h>
#include "FreeRTOS.h"
#include "task.h"
#include "stream_buffer.h"
```

### Data Structures

- union [weigand26\\_frame\\_t](#)
- struct [weigand26\\_buff\\_item\\_t](#)

### Macros

- #define **WEIGAND26\_FRAME\_SIZE** 26
- #define **WEIGAND26\_FRAME\_TIME\_LIMIT** 80
- #define **WEIGAND26\_BUFF\_ITEM\_SIZE** sizeof([weigand26\\_buff\\_item\\_t](#))

### Functions

- void [weigand\\_init](#) (StreamBufferHandle\_t buffer, uint8\_t id, uint8\_t dx\_port, uint8\_t d0\_pin, uint8\_t d1\_pin)
   
*Initialize Wiegand driver.*
- void [weigand\\_disable](#) (uint8\_t dx\_port, uint8\_t d0\_pin, uint8\_t d1\_pin)
   
*Disable Wiegand driver.*
- uint32\_t [weigand\\_get\\_facility](#) ([weigand26\\_frame\\_t](#) frame)
   
*Parse facility code from frame.*
- uint32\_t [weigand\\_get\\_card](#) ([weigand26\\_frame\\_t](#) frame)
   
*Parse card number from frame.*
- bool [weigand\\_is\\_parity\\_ok](#) ([weigand26\\_frame\\_t](#) frame)
   
*Check frame parity.*

### 4.20.1 Detailed Description

Wiegand26 interface driver.

500 b/s transfer rate data pulse 40-70us data interval >2ms

Only Wiegand 26bit format Frame: | even parity (1b) | facility code (8b) | card number (16b) | odd parity (1b) | transmission duration ~52ms

#### Author

Petr Elexa

#### See also

LICENSE

### 4.20.2 Function Documentation

#### 4.20.2.1 weigand\_disable()

```
void weigand_disable (
 uint8_t dx_port,
 uint8_t d0_pin,
 uint8_t d1_pin)
```

Disable Wiegand driver.

#### Parameters

|                |                                   |
|----------------|-----------------------------------|
| <i>dx_port</i> | ... Port number for data signals. |
| <i>d0_pin</i>  | ... Pin for 0's data signal.      |
| <i>d1_pin</i>  | ... Pin for 1's data signal.      |

#### 4.20.2.2 weigand\_get\_card()

```
uint32_t weigand_get_card (
 weigand26_frame_t frame)
```

Parse card number from frame.

#### Parameters

|              |                           |
|--------------|---------------------------|
| <i>frame</i> | ... Wiegand26 data frame. |
|--------------|---------------------------|

**Returns**

card number

**4.20.2.3 weigand\_get\_facility()**

```
uint32_t weigand_get_facility (
 weigand26_frame_t frame)
```

Parse facility code from frame.

**Parameters**

|              |                           |
|--------------|---------------------------|
| <i>frame</i> | ... Wiegand26 data frame. |
|--------------|---------------------------|

**Returns**

facility code

**4.20.2.4 weigand\_init()**

```
void weigand_init (
 StreamBufferHandle_t buffer,
 uint8_t id,
 uint8_t dx_port,
 uint8_t d0_pin,
 uint8_t d1_pin)
```

Initialize Wiegand driver.

**Note**

One buffer for each consumer is preferred.

**Parameters**

|                |                                                                                              |
|----------------|----------------------------------------------------------------------------------------------|
| <i>buffer</i>  | ... Receive buffer for frames from RFID card/tags. See <a href="#">weigand26_buff_item_t</a> |
| <i>id</i>      | ... interface identification                                                                 |
| <i>dx_port</i> | ... Port number for data signals.                                                            |
| <i>d0_pin</i>  | ... Pin for 0's data signal.                                                                 |
| <i>d1_pin</i>  | ... Pin for 1's data signal.                                                                 |

#### 4.20.2.5 `weigand_is_parity_ok()`

```
bool weigand_is_parity_ok (
 weigand26_frame_t frame)
```

Check frame parity.

##### Parameters

|              |                           |
|--------------|---------------------------|
| <i>frame</i> | ... Wiegand26 data frame. |
|--------------|---------------------------|

##### Returns

true if parity is valid



# Index

\_check\_system\_stack\_size  
    start.c, 25

\_timing\_calculate  
    can\_term\_driver.c, 14

acs\_can\_protocol.h, 9

acs\_msg\_data\_auth\_req\_t, 5

acs\_msg\_data\_auth\_resp\_t, 5

acs\_msg\_data\_door\_ctrl\_t, 5

acs\_msg\_head\_t, 6

BOD\_Init  
    brownout.c, 11  
    brownout.h, 12

BOD\_IRQHandler  
    brownout.c, 11  
    brownout.h, 12

brownout.c, 10  
    BOD\_Init, 11  
    BOD\_IRQHandler, 11

brownout.h, 11  
    BOD\_Init, 12  
    BOD\_IRQHandler, 12

cache\_item\_t, 6

CAN\_init  
    can\_term\_driver.c, 14  
    can\_term\_driver.h, 17

CAN\_IRQHandler  
    can\_term\_driver.c, 14  
    can\_term\_driver.h, 17

CAN\_recv\_filter  
    can\_term\_driver.c, 14  
    can\_term\_driver.h, 17

CAN\_recv\_filter\_all\_ext  
    can\_term\_driver.c, 15  
    can\_term\_driver.h, 18

CAN\_send\_once  
    can\_term\_driver.c, 15  
    can\_term\_driver.h, 18

CAN\_send\_test  
    can\_term\_driver.c, 15  
    can\_term\_driver.h, 18

can\_term\_driver.c, 12  
    \_timing\_calculate, 14  
    CAN\_init, 14  
    CAN\_IRQHandler, 14  
    CAN\_recv\_filter, 14  
    CAN\_recv\_filter\_all\_ext, 15  
    CAN\_send\_once, 15

    CAN\_send\_test, 15

can\_term\_driver.h, 16  
    CAN\_init, 17  
    CAN\_IRQHandler, 17  
    CAN\_recv\_filter, 17  
    CAN\_recv\_filter\_all\_ext, 18  
    CAN\_send\_once, 18  
    CAN\_send\_test, 18

get\_reader\_a\_addr  
    terminal\_config.c, 40  
    terminal\_config.h, 43

get\_reader\_b\_addr  
    terminal\_config.c, 40  
    terminal\_config.h, 43

I2C\_IRQHandler  
    storage.c, 30

main  
    start.c, 25

map\_reader\_idx\_to\_cache  
    terminal.c, 35

reader.c, 19  
    reader\_conf, 21  
    reader\_deinit, 20  
    reader\_get\_request\_from\_buffer, 20  
    reader\_init, 20  
    reader\_is\_door\_open, 21  
    reader\_unlock, 21

reader.h, 22  
    reader\_deinit, 23  
    reader\_get\_request\_from\_buffer, 23  
    reader\_init, 24  
    reader\_is\_door\_open, 24  
    reader\_unlock, 24

reader\_conf  
    reader.c, 21

reader\_conf\_t, 7

reader\_deinit  
    reader.c, 20  
    reader.h, 23

reader\_get\_request\_from\_buffer  
    reader.c, 20  
    reader.h, 23

reader\_init  
    reader.c, 20  
    reader.h, 24

reader\_is\_door\_open

reader.c, 21  
reader.h, 24  
reader\_unlock  
  reader.c, 21  
  reader.h, 24  
reader\_wiring\_t, 7  
  
set\_reader\_addr  
  terminal\_config.c, 40  
  terminal\_config.h, 43  
start.c, 24  
  \_check\_system\_stack\_size, 25  
  main, 25  
static\_cache.c, 26  
static\_cache.h, 26  
  static\_cache\_convert, 27  
  static\_cache\_erase, 28  
  static\_cache\_get, 28  
  static\_cache\_insert, 28  
  static\_cache\_reset, 29  
  STATIC\_CACHE\_SETS, 27  
static\_cache\_convert  
  static\_cache.h, 27  
static\_cache\_erase  
  static\_cache.h, 28  
static\_cache\_get  
  static\_cache.h, 28  
static\_cache\_insert  
  static\_cache.h, 28  
static\_cache\_reset  
  static\_cache.h, 29  
STATIC\_CACHE\_SETS  
  static\_cache.h, 27  
storage.c, 29  
  I2C\_IRQHandler, 30  
  storage\_init, 30  
  storage\_read\_byte, 30  
  storage\_read\_word\_le, 30  
  storage\_write\_byte, 31  
  storage\_write\_word\_le, 31  
storage.h, 31  
  storage\_init, 32  
  storage\_read\_byte, 32  
  storage\_read\_word\_le, 33  
  storage\_write\_byte, 33  
  storage\_write\_word\_le, 33  
storage\_init  
  storage.c, 30  
  storage.h, 32  
storage\_read\_byte  
  storage.c, 30  
  storage.h, 32  
storage\_read\_word\_le  
  storage.c, 30  
  storage.h, 33  
storage\_write\_byte  
  storage.c, 31  
  storage.h, 33  
storage\_write\_word\_le

storage.c, 31  
storage.h, 33  
  
term\_can\_error  
  terminal.c, 35  
  terminal.h, 37  
term\_can\_recv  
  terminal.c, 35  
  terminal.h, 38  
term\_can\_send  
  terminal.c, 36  
  terminal.h, 38  
terminal.c, 34  
  map\_reader\_idx\_to\_cache, 35  
  term\_can\_error, 35  
  term\_can\_recv, 35  
  term\_can\_send, 36  
  terminal\_init, 36  
  terminal\_reconfigure, 36  
  USER\_REQUEST\_WAIT\_MS, 37  
terminal.h, 37  
  term\_can\_error, 37  
  term\_can\_recv, 38  
  term\_can\_send, 38  
  terminal\_init, 38  
  terminal\_reconfigure, 39  
terminal\_config.c, 39  
  get\_reader\_a\_addr, 40  
  get\_reader\_b\_addr, 40  
  set\_reader\_addr, 40  
  terminal\_config\_init, 40  
terminal\_config.h, 41  
  get\_reader\_a\_addr, 43  
  get\_reader\_b\_addr, 43  
  set\_reader\_addr, 43  
  terminal\_config\_init, 43  
terminal\_config\_init  
  terminal\_config.c, 40  
  terminal\_config.h, 43  
terminal\_init  
  terminal.c, 36  
  terminal.h, 38  
terminal\_reconfigure  
  terminal.c, 36  
  terminal.h, 39  
  
USER\_REQUEST\_WAIT\_MS  
  terminal.c, 37  
  
watchdog.c, 44  
  WDT\_Feed, 45  
  WDT\_Init, 45  
  WDT\_IRQHandler, 45  
watchdog.h, 45  
  WDT\_Feed, 46  
  WDT\_Init, 46  
  WDT\_IRQHandler, 47  
WDT\_Feed  
  watchdog.c, 45

- watchdog.h, 46
- WDT\_Init
  - watchdog.c, 45
  - watchdog.h, 46
- WDT\_IRQHandler
  - watchdog.c, 45
  - watchdog.h, 47
- weigand.c, 47
  - weigand\_disable, 48
  - weigand\_get\_card, 48
  - weigand\_get\_facility, 49
  - weigand\_init, 49
  - weigand\_is\_parity\_ok, 50
- weigand.h, 50
  - weigand\_disable, 51
  - weigand\_get\_card, 51
  - weigand\_get\_facility, 52
  - weigand\_init, 52
  - weigand\_is\_parity\_ok, 52
- weigand26\_buff\_item\_t, 7
- weigand26\_frame\_t, 8
- weigand26\_t, 8
- weigand\_disable
  - weigand.c, 48
  - weigand.h, 51
- weigand\_get\_card
  - weigand.c, 48
  - weigand.h, 51
- weigand\_get\_facility
  - weigand.c, 49
  - weigand.h, 52
- weigand\_init
  - weigand.c, 49
  - weigand.h, 52
- weigand\_is\_parity\_ok
  - weigand.c, 50
  - weigand.h, 52