Vehicle Control Unit (VCU)

Firmware Specifications

History

Version	Author	Date
1.0	Arella Matteo	2018

Contents

Chapter 1

FastChargeSAE VCU firmware

VCU is an embedded system employeed to control the functional operations of the vehicle. It takes care of the management of the state of drive according to the Finite State Machine (FSM) section.

Chapter 2

CAN Networks

Two CAN networks have been designed to be inserted into the vehicle: a first CAN network between the VCU and the inverter (CAN funzionale) and a second CAN network between the VCU, TCU and SCUs (CAN servizi).

Each node connected to CAN network has an unique ID internally to that network, according to this table:

CAN NETWORK	NODE	NODE-ID
FUNZIONALE	INVERTER	1
	VCU	2
SERVIZI	FrontalSCU	1
	VCU	2
	TCU	4

- CAN funzionale
- CAN servizi

2.1 CAN Funzionale

CAN funzionale network arises from the need to have an high reliable and robust communication between VCU and inverter.

VCU master on power up sequence

1. **wait for BOOT-UP message from inverter**

COB-ID (11bits)	data byte 0
0x700 + NODE-ID	0x00

2. **check VENDOR-ID** (VCU read object 0x1018)
   VCU send cliend SDO

COB-ID (11bits)	Command byte	Obj. Index (2 byte)	Obj. sub-index (byte)	Data (4bytes)
0x600 + NODE-ID	0x40	0x1810	0x01	0

VCU receive server SDO (from inverter)

COB-ID (11bits)	Command byte	Obj. Index (2 byte)	Obj. sub-index (byte)	Data (4bytes)
0x580 + NODE-ID	0x43	0x1810	0x01	0x19030000

3. **send NMT 'go Operational'** (broadcast)

COB-ID (11bits)	data byte 0	data byte 1
0x000	0x01	0x00

4. **send PDO to enable PWM**

COB-ID (11bits)	Data (RPDO1)
0x200 + NODE-ID	06 00 xx xx xx xx xx xx

COB-ID (11bits)	Data (RPDO1)
0x200 + NODE-ID	07 00 xx xx xx xx xx xx

COB-ID (11bits)	Data (RPDO1)
0x200 + NODE-ID	0F 00 xx xx xx xx xx xx

5. **periodically send SYNC message**
VCU broadcast SYNC

COB-ID (11bits)	Data byte 0
0x80 + NODE-ID	0x00

2.2 CAN Servizi

CAN servizi network arises from the need to digitize all those signals necessary for the operation of the car. VCU master node is interested to receive pedals position percentage values, APPS and brake plausibility from frontal SCU, and torque request limiter from TCU.

A CANbus communication protocol has been developed based on the CANopen specifications (CiA 301). Each slave node can be represented by a finite state machine with the following states: Initialisation, Pre-operational, Operational, Stopped. During power-up each node is in the Initialization phase. At the end of this phase, it attempts to send a boot-up message. As soon as it has been successfully sent, it is placed in the Pre-operational state. Using an NMT master message, the VCU can passes the various slave nodes between the different Pre-operational, Operational and Stopped states.

VCU master on power up sequence

1. **wait for BOOT-UP message from slave nodes** (from frontal SCU and TCU)

COB-ID (11bits)	data byte 0
0x700 + NODE-ID	0x00

2. **send NMT 'go Operational'** (broadcast)

COB-ID (11bits)	data byte 0	data byte 1
0x000	0x01	0x00

Fault Tolerance

If failure of CAN servizi network occurs, redundancy has been introduced in the acquisition of pedals: if PDO with the pedal data are not received within a certain timeout then CAN servizi network is considered non-functional and pedal sensors are acquired directly via analog signals from the VCU.

Chapter 3

Board model

VCU is based on an Atmel SAM3X8E board with an ARM Cortex-M3 microprocessor.

IO signals managed from VCU are:

- ANALOG
  1. First APPS
  2. Second APPS
  3. Brake pedal position sensor
  4. SC voltage
- DIGITAL
  1. AIR+
  2. AIR-
  3. Precharge
  4. RTDS buzzer
  5. AIR button
  6. RTD button

In case of CAN Servizi failure both APPS and brake pedal position sensor are acquired by analog signals for fault tolerance.

Chapter 4

Ready To Drive Sound (RTDS)

The vehicle must make a characteristic sound, continuously for at least one second and a maximum of three seconds when it enters ready-to-drive mode. According Finite State Machine (FSM), RTDS is triggered with transition from HVON state to DRIVE state.

Chapter 5

Finite State Machine (FSM)

VCU firmware is based upon 4 states:

- STAND: ignition of the car, you return here every time SC is undervoltage
- HV ON: active high voltage is only accessed from STAND via AIRbutton and SC voltage $>$ 3V
- DRIVE: the safe driving status, accessible by RTD procedure but also with pedals implausibility through the return procedure
- NOT DRIVE: error with pedals, the sensors of the pedals are disconnected or out range. Torque request and braking request are disabled; you enter only by pedals implausibility or absence of the brake.

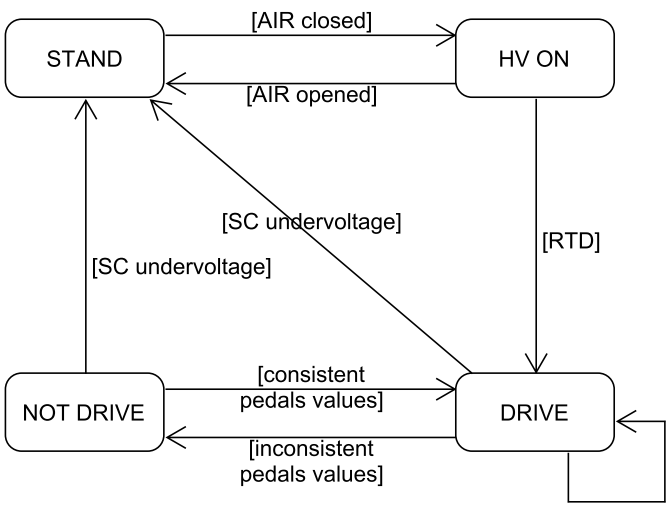


Figure 5.1 FSM diagram

Chapter 6

Module Index

6.1 Modules

Here is a list of all modules:

Chapter 7

File Index

7.1 File List

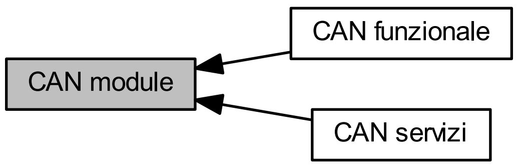
Here is a list of all documented files with brief descriptions:

Chapter 8

Module Documentation

8.1 CAN module

Collaboration diagram for CAN module:



Modules

- CAN funzionale
- CAN servizi

Macros

- #define VCU_NODE_ID 2

  VCU Node ID.

Functions

- bool can_init ()

  This function initializes both CAN funzionale and CAN servizi networks.

8.1.1 Detailed Description

8.1.2 Function Documentation

8.1.2.1 can_init()

```
bool can_init ( )
```

This function initializes both CAN funzionale and CAN servizi networks.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

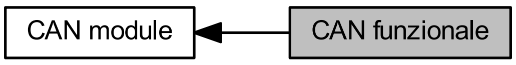
Return values

true	CAN networks initialized successfully
false	CAN networks initialization failed

Definition at line 26 of file CO_can.cpp.

8.2 CAN funzionale

Collaboration diagram for CAN funzionale:



Macros

- #define INVERTER_NODE_ID 1

  Inverter Node ID.

Functions

- volatile bool can_funzionale_initialized ()

  This function returns CAN funzionale initialization status.
- void can_funzionale_send_sync ()

  This function sends a periodic CANOpen sync message to inverter slave node.
- void CAN_FUNZ_BOOTUP_CB (CAN_FRAME ∗frame)

  This function manage boot-up message sent over CAN funzionale network by inverter slave node. Upon boot-up message reception the VCU send a SDO client request for check inverter vendor ID; then inverter is considered online over CAN funzionale network.
- void CAN_FUNZ_VENDOR_ID_CB (CAN_FRAME ∗frame)

  This function manage SDO server response with inverter Vendor ID. VCU sends NMT operational and PDOs to enable PWM; then inverter is considered correctly configured and a timer is started for sending periodic sync messages.
- void CAN_FUNZ_GENERAL_CB (CAN_FRAME ∗frame)

  This function manage TPDO from inverter and deserializes data:
- bool can_funzionale_init ()

  This function initialize CAN funzionale hardware port with baudrate CAN_FUNZ_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from inverter slave node (filter = 0x00000700 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); mailbox 1 is configured for receiving vendorID SDO response from inverter (filter = 0x00000580 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); remaining mailboxes are configured for receiving TPDOs from inverter slave node (filter = 0x00000080, mask = 0x1FFFFCFF).
- volatile bool can_funzionale_online ()

  This function returns if inverter is online and active over CAN funzionale.
- void inverter_torque_request (uint16_t torque)

  This function send torque request to inverter. If inverter is active over CAN funzionale network then the request is done via RPDO1 viceversa it's done via analog signal.
- void inverter_regen_request (uint16_t regen)

  This function send regen request to inverter.
- volatile uint16_t get_torque_actual_value ()

  This function return the torque value requested by inverter to motor retrieved from TPDO1 from inverter over CAN funzionale network.

Variables

- volatile bool can_funz_initialized = false

  CAN funzionale initialization status flag (true if initialized)
- volatile bool inverter_online = false

  Inverter online status flag (true if online)
- volatile bool inverter_configured = false

  Inverter configured status flag (true if configured)
- volatile uint16_t torque_actual_value = 0

  Torque requested by inverter to motor.

8.2.1 Detailed Description

8.2.2 Function Documentation

8.2.2.1 CAN_FUNZ_BOOTUP_CB()

```
void CAN_FUNZ_BOOTUP_CB (
             CAN_FRAME * frame )
```

This function manage boot-up message sent over CAN funzionale network by inverter slave node. Upon boot-up message reception the VCU send a SDO client request for check inverter vendor ID; then inverter is considered online over CAN funzionale network.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Parameters

in	*frame*	CAN frame received from CAN funzionale port

Definition at line 77 of file can_funzionale.cpp.

8.2.2.2 CAN_FUNZ_GENERAL_CB()

```
void CAN_FUNZ_GENERAL_CB (
             CAN_FRAME * frame )
```

This function manage TPDO from inverter and deserializes data:

TPDO num	NODE-ID	Data
1	INVERTER_NODE_ID	Torque Actual Val

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Parameters

in	*frame*	CAN frame received from CAN servizi port

Definition at line 170 of file can_funzionale.cpp.

8.2.2.3 CAN_FUNZ_VENDOR_ID_CB()

```
void CAN_FUNZ_VENDOR_ID_CB (
             CAN_FRAME * frame )
```

This function manage SDO server response with inverter Vendor ID. VCU sends NMT operational and PDOs to enable PWM; then inverter is considered correctly configured and a timer is started for sending periodic sync messages.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Parameters

in	*frame*	CAN frame received from CAN servizi port

Definition at line 108 of file can_funzionale.cpp.

8.2.2.4 can_funzionale_init()

```
bool can_funzionale_init ( )
```

This function initialize CAN funzionale hardware port with baudrate CAN_FUNZ_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from inverter slave node (filter = 0x00000700 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); mailbox 1 is configured for receiving vendorID SDO response from inverter (filter = 0x00000580 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); remaining mailboxes are configured for receiving TPDOs from inverter slave node (filter = 0x00000080, mask = 0x1FFFFCFF).

This function initializes CAN funzionale network with inverter.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Return values

true	CAN servizi initialized
false	CAN servizi not initialized

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Return values

true	CAN funzionale network initialized successfully
false	CAN funzionale network initialization failed

Definition at line 192 of file can_funzionale.cpp.

8.2.2.5 can_funzionale_initialized()

```
volatile bool can_funzionale_initialized ( )
```

This function returns CAN funzionale initialization status.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Return values

true	CAN funzionale network initialized
false	CAN funzionale network not initialized

Definition at line 44 of file can_funzionale.cpp.

8.2.2.6 can_funzionale_online()

```
volatile bool can_funzionale_online ( )
```

This function returns if inverter is online and active over CAN funzionale.

This function returns if CAN funzionale network is online.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Return values

true	CAN funzionale initialized, inverter online and inverter configured successfully
false	CAN funzionale not initialized or inverter not online or configured.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Return values

true	CAN funzionale network online
false	CAN funzionale network offline

Definition at line 226 of file can_funzionale.cpp.

8.2.2.7 can_funzionale_send_sync()

```
void can_funzionale_send_sync ( )
```

This function sends a periodic CANOpen sync message to inverter slave node.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 55 of file can_funzionale.cpp.

8.2.2.8 get_torque_actual_value()

```
volatile uint16_t get_torque_actual_value ( )
```

This function return the torque value requested by inverter to motor retrieved from TPDO1 from inverter over CAN funzionale network.

This function return the torque value requested by inverter to motor.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Returns

Torque requested by inverter to motor

Definition at line 276 of file can_funzionale.cpp.

8.2.2.9 inverter_regen_request()

```
void inverter_regen_request (
            uint16_t regen )
```

This function send regen request to inverter.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Parameters

in	*regen*	Regen request value

Definition at line 258 of file can_funzionale.cpp.

8.2.2.10 inverter_torque_request()

```
void inverter_torque_request (
            uint16_t torque )
```

This function send torque request to inverter. If inverter is active over CAN funzionale network then the request is done via RPDO1 viceversa it's done via analog signal.

This function send torque request to inverter.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

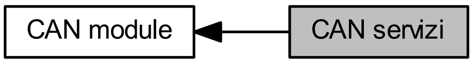
Parameters

in	*torque*	Torque value in percentage multiplied per tcs torque limiter coefficient

Definition at line 241 of file can_funzionale.cpp.

8.3 CAN servizi

Collaboration diagram for CAN servizi:



Macros

- #define SCU_FRONTAL_NODE_ID 1

  Frontal SCU Node ID.
- #define TCU_NODE_ID 4

  TCU Node ID.

Functions

- void timeout ()

  This function is executed periodically after CAN servizi 'go Operational' NMT request is sent. When timeout occurs if next_pedals_seq_num is greater than curr_pedals_seq_num then frontal SCU is considered active, viceversa it is considered offline.
- volatile bool can_servizi_initialized ()

  This function returns CAN servizi initialization status.
- void CAN_SERV_BOOTUP_CB (CAN_FRAME ∗frame)

  This function manage boot-up messages sent over CAN servizi network by slave nodes.
- void CAN_SERV_GENERAL_CB (CAN_FRAME ∗frame)

  This function manage PDOs received over CAN servizi network and deserializes data:
- bool can_servizi_init ()

  This function initialize CAN servizi hardware port with baudrate CAN_SERV_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from CAN servizi slave nodes (filter = 0x00000700, mask = 0x1FFFFF80); remaining mailboxes are configured for receiving TPDOs from CAN servizi slave nodes (filter = 0x00000080, mask = 0x1FFF↩ FC80).
- void can_servizi_go_operational ()

  This function send a CANOpen master NMT message for request 'go to Operational' state to CAN servizi slave nodes (SCUs and TCU).
- volatile bool can_servizi_online ()

  This function returns if CAN servizi network is online.
- volatile bool tcs_online ()

  This function returns if TCU node is active and online on the CAN servizi network.
- volatile uint8_t get_servizi_tps1 ()

  This function returns the value of the first APPS in percentage, retrieved by frontal SCU node over CAN servizi network.
- volatile uint8_t get_servizi_tps2 ()

  This function returns the value of the second APPS in percentage, retrieved by frontal SCU node over CAN servizi network.

- volatile uint8_t get_servizi_brake ()

  This function returns the value of brake pedal position sensor in percentage, retrieved by frontal SCU node over CAN servizi network.
- volatile bool get_servizi_apps_plausibility ()

  This function returns the value of APPS plausibility retrieved by frontal SCU node over CAN servizi network.
- volatile bool get_servizi_brake_plausibility ()

  This function returns the value of brake plausibility retrieved by frontal SCU node over CAN servizi network.
- volatile uint8_t get_tcs_torque_coefficient ()

  This function returns the value of torque limiter percentage retrieved by TCU node over CAN servizi network.

Variables

- volatile bool can_serv_initialized = false

  CAN servizi initialization status flag (true if initialized)
- volatile bool SCU_F_online = false

  Frontal SCU online status flag (true if online)
- volatile bool TCS_online = false

  TCS online status flag (true if online)
- volatile uint32_t curr_pedals_seq_num = 0

  Frontal SCU PDOtx1 current sequence number.
- volatile uint32_t next_pedals_seq_num = 0

  Frontal SCU PDOtx1 next sequence number.
- volatile uint8_t tps1_percentage = 0

  First APPS percentage value retrieved by frontal SCU node.
- volatile uint8_t tps2_percentage = 0

  Second APPS percentage value retrieved by frontal SCU node.
- volatile uint8_t brake_percentage = 0

  Brake pedal position sensor percentage value retrieved by frontal SCU node.
- volatile bool apps_plausibility = true

  APPS plausibility status retrieved by frontal SCU node.
- volatile bool brake_plausibility = true

  Brake plausibility status retrieved by frontal SCU node.
- volatile uint8_t tcs_coefficient = 0

  torque limiter percentage retrieved by TCU node

8.3.1 Detailed Description

8.3.2 Function Documentation

8.3.2.1 CAN_SERV_BOOTUP_CB()

```
void CAN_SERV_BOOTUP_CB (
             CAN_FRAME * frame )
```

This function manage boot-up messages sent over CAN servizi network by slave nodes.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Parameters

in	*frame*	CAN frame received from CAN servizi port

Definition at line 115 of file can_servizi.cpp.

8.3.2.2 CAN_SERV_GENERAL_CB()

```
void CAN_SERV_GENERAL_CB (
             CAN_FRAME * frame )
```

This function manage PDOs received over CAN servizi network and deserializes data:

TPDO num	NODE-ID	Length	Data
1	SCU_FRONTAL_NODE_ID	4	APPS1 percentage
			APPS2 percentage
			Brake percentage
			APPS plausibility
			BRAKE plausibility
1	TCU_NODE_ID	1	TCU torque limiter

When PDOtx1 message is received from frontal SCU node then next_pedals_seq_num is incremented for keep track of last pedals message received.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Parameters

in	*frame*	CAN frame received from CAN servizi port

Definition at line 149 of file can_servizi.cpp.

8.3.2.3 can_servizi_go_operational()

```
void can_servizi_go_operational ( )
```

This function send a CANOpen master NMT message for request 'go to Operational' state to CAN servizi slave nodes (SCUs and TCU).

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Definition at line 206 of file can_servizi.cpp.

8.3.2.4 can_servizi_init()

```
bool can_servizi_init ( )
```

This function initialize CAN servizi hardware port with baudrate CAN_SERV_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from CAN servizi slave nodes (filter = 0x00000700, mask = 0x1FFFFF80); remaining mailboxes are configured for receiving TPDOs from CAN servizi slave nodes (filter = 0x00000080, mask = 0x1FFFFC80).

This function initializes CAN servizi network.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Return values

true	CAN servizi initialized
false	CAN servizi not initialized

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Return values

true	CAN servizi network initialized successfully
false	CAN servizi network initialization failed

Definition at line 182 of file can_servizi.cpp.

8.3.2.5 can_servizi_initialized()

```
volatile bool can_servizi_initialized ( )
```

This function returns CAN servizi initialization status.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Return values

true	CAN servizi network initialized
false	CAN servizi network not initialized

Definition at line 102 of file can_servizi.cpp.

8.3.2.6 can_servizi_online()

```
volatile bool can_servizi_online ( )
```

This function returns if CAN servizi network is online.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Return values

true	CAN servizi network online
false	CAN servizi network offline

Definition at line 220 of file can_servizi.cpp.

8.3.2.7 get_servizi_apps_plausibility()

```
volatile bool get_servizi_apps_plausibility ( )
```

This function returns the value of APPS plausibility retrieved by frontal SCU node over CAN servizi network.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Return values

true	APPS plausibility
false	APPS implausibility

Definition at line 245 of file can_servizi.cpp.

8.3.2.8 get_servizi_brake()

```
volatile uint8_t get_servizi_brake ( )
```

This function returns the value of brake pedal position sensor in percentage, retrieved by frontal SCU node over CAN servizi network.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Returns

Brake pedal position percentage value

Definition at line 240 of file can_servizi.cpp.

8.3.2.9 get_servizi_brake_plausibility()

```
volatile bool get_servizi_brake_plausibility ( )
```

This function returns the value of brake plausibility retrieved by frontal SCU node over CAN servizi network.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Return values

true	Brake plausibility
false	Brake implausibility

Definition at line 250 of file can_servizi.cpp.

8.3.2.10 get_servizi_tps1()

```
volatile uint8_t get_servizi_tps1 ( )
```

This function returns the value of the first APPS in percentage, retrieved by frontal SCU node over CAN servizi network.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Returns

First APPS percentage value

Definition at line 230 of file can_servizi.cpp.

8.3.2.11 get_servizi_tps2()

```
volatile uint8_t get_servizi_tps2 ( )
```

This function returns the value of the second APPS in percentage, retrieved by frontal SCU node over CAN servizi network.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Returns

Second APPS percentage value

Definition at line 235 of file can_servizi.cpp.

8.3.2.12 get_tcs_torque_coefficient()

```
volatile uint8_t get_tcs_torque_coefficient ( )
```

This function returns the value of torque limiter percentage retrieved by TCU node over CAN servizi network.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Returns

Torque limiter coefficient in percentage

Definition at line 255 of file can_servizi.cpp.

8.3.2.13 tcs_online()

```
volatile bool tcs_online ( )
```

This function returns if TCU node is active and online on the CAN servizi network.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Return values

true	TCS is online and active
false	TCS is offline

Definition at line 225 of file can_servizi.cpp.

8.3.2.14 timeout()

```
void timeout ( )
```

This function is executed periodically after CAN servizi 'go Operational' NMT request is sent. When timeout occurs if next_pedals_seq_num is greater than curr_pedals_seq_num then frontal SCU is considered active, viceversa it is considered offline.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 93 of file can_servizi.cpp.

8.4 Filter buffer

Macros

- #define USE_LOOP_UNROLLING (1)

  Flag macro for using or not loop unrolling into filter function.
- #define pos(x, offset) ((x) ∗ offset)

  Buffer indexing macro.

Functions

- uint16_t filter_buffer (volatile uint16_t ∗buffer, int size, unsigned offset)

  This function filters the input buffer with an average filter.

8.4.1 Detailed Description

8.4.2 Function Documentation

8.4.2.1 filter_buffer()

```
uint16_t filter_buffer (
            volatile uint16_t * buffer,
            int size,
            unsigned offset )
```

This function filters the input buffer with an average filter.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Parameters

in	*buffer*	Input buffer
in	*size*	Buffer size
in	*offset*	Offset between data corresponding to same acquired value

Returns

Filtered data

Definition at line 39 of file filter.cpp.

8.5 Board model

Macros

- #define CAN_FUNZIONALE Can1

  Pin on board dedicated to CAN funzionale port.
- #define CAN_SERVIZI Can0

  Pin on board dedicated to CAN servizi port.
- #define AIRcc 48

  Pin on board dedicated to AIR+.
- #define AIRGnd 49

  Pin on board dedicated to AIR-.
- #define PRE 47

  Pin on board dedicated to PRECHARGE.
- #define BUZZER 52

  Pin on board dedicated to buzzer for RTDS.
- #define AIRB 14

  Pin on board dedicated to AIR button.
- #define RTDB 15

  Pin on board dedicated to RTD button.
- #define FAN 9

  Pin on board dedicated to FAN.
- #define INVERTER_ANALOG_PIN DAC1

  Pin on board dedicated to inverter torque request analog signal.
- #define BRAKE_REGEN_PIN DAC0

  Pin on board dedicated to inverter regen request analog signal.
- #define TPS1_PIN A0

  Pin on board dedicated to first APPS (tps1)
- #define TPS1_ADC_CHAN_NUM ADC_CHER_CH7

  GPIO pin on the Atmel SAM3X8E processor corresponding to tps1 signal (AD7)
- #define TPS2_PIN A1

  Pin on board dedicated to second APPS (tps2)
- #define TPS2_ADC_CHAN_NUM ADC_CHER_CH6

  GPIO pin on the Atmel SAM3X8E processor corresponding to tps2 signal (AD6)
- #define BRAKE_PIN A2

  Pin on board dedicated to brake pedal position sensor.
- #define BRAKE_ADC_CHAN_NUM ADC_CHER_CH5

  GPIO pin on the Atmel SAM3X8E processor corresponding to brake signal (AD5)
- #define SC_PIN A3

  Pin on board dedicated to SC read signal.
- #define SC_ADC_CHAN_NUM ADC_CHER_CH4

  GPIO pin on the Atmel SAM3X8E processor corresponding to SC signal (AD4)
- #define ADC_BUFFER_SIZE 128

  Size (bytes) of buffer for store each ADC channel data with DMA.
- #define BUFFERS 4

  Number of DMA buffers.
- #define ADC_MIN 0

  ADC lower bound value.
- #define ADC_MAX 4095

  ADC upper bound value.

- #define ADC_CHANNELS_LIST TPS1_ADC_CHAN_NUM | TPS2_ADC_CHAN_NUM | BRAKE_ADC_CHAN_NUM | SC_ADC_CHAN_NUM

  List of ADC channels dedicated to each IO port.
- #define ADC_CHANNELS 4

  Number of ADC channels.
- #define TPS1_ADC_OFFSET 0

  Offset from DMA buffer.
- #define TPS2_ADC_OFFSET 1

  Offset from DMA buffer.
- #define BRAKE_ADC_OFFSET 2

  Offset from DMA buffer.
- #define SC_ADC_OFFSET 3

  Offset from DMA buffer.
- #define BUFFER_LENGTH ADC_BUFFER_SIZE ∗ ADC_CHANNELS

  Length, in bytes, of each DMA buffer.
- #define APPS_PLAUS_RANGE 10

  Size (bytes) of each ADC buffer.
- #define TPS1_UPPER_BOUND 2482

  First APPS max output voltage (2V)
- #define TPS1_LOWER_BOUND 993

  First APPS min output voltage (0.8V)
- #define TPS2_UPPER_BOUND 1241

  Second APPS max output voltage (1V)
- #define TPS2_LOWER_BOUND 497

  Second APPS min output voltage (0.4V)
- #define BRAKE_UPPER_BOUND 0

  Brake sensor max output voltage (TODO: check Voutmax)
- #define BRAKE_LOWER_BOUND ADC_MAX

  Brake sensor min output voltage (TODO: check Voutmin)

Functions

- void ADC_Handler ()

  ADC IRQ handler. When ADC buffer is filled DMA pointer is linked to next buffer available. Then acquired data are filtered.
- void model_init ()

  This function initializes hardware board. Both APPS and brake pedal position sensor are acquired by analog signals for fault tolerance in case of CAN servizi fault.
- volatile uint8_t get_tps1_percentage ()

  This function returns the value of the first APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint8_t get_tps2_percentage ()

  This function returns the value of the second APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint8_t get_brake_percentage ()

  This function returns the value of the brake pedal position sensor in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile bool get_apps_plausibility ()

  This function returns the value of APPS plausibility retrieved by CAN servizi network, if online, or by analog signal.
- volatile bool get_brake_plausibility ()

  This function returns the value of brake plausibility retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint16_t get_SC_value ()

  This function returns the value of the SC.

Variables

- volatile uint8_t tps1_adc_percentage = 0

  First APPS percentage value retrieved by analog tps1 signal (TPS1_PIN)
- volatile uint8_t tps2_adc_percentage = 0

  Second APPS percentage value retrieved by analog tps2 signal (TPS2_PIN)
- volatile uint8_t brake_adc_percentage = 0

  Brake pedal position sensor percentage value retrieved by analog brake signal (BRAKE_PIN)
- volatile bool apps_adc_plausibility = true

  APPS plausibility status retrieved by analog acquisition.
- volatile bool brake_adc_plausibility = true

  Brake plausibility status retrieved by analog acquisition.
- volatile uint16_t tps1_value = 0

  First APPS value retrieved directly by analog tps1 signal (TPS1_PIN) and filtered after DMA buffer is filled entirely.
- volatile uint16_t tps2_value = 0

  Second APPS value retrieved directly by analog tps2 signal (TPS2_PIN) and filtered after DMA buffer is filled entirely.
- volatile uint16_t brake_value = 0

  Brake pedal position sensor value retrieved directly by analog brake signal (BRAKE_PIN) and filtered after DMA buffer is filled entirely.
- volatile uint16_t SC_value = 0

  SC value retrieved directly by analog SC signal (SC_PIN) and filtered after DMA buffer is filled entirely.
- volatile uint16_t tps1_max = 2482

  First APPS max output voltage (equals to TPS1_UPPER_BOUND)
- volatile uint16_t tps1_min = 993

  First APPS min output voltage (equals to TPS1_LOWER_BOUND)
- volatile uint16_t tps2_max = 1241

  Second APPS max output voltage (equals to TPS2_UPPER_BOUND)
- volatile uint16_t tps2_min = 497

  Second APPS min output voltage (equals to TPS2_LOWER_BOUND)
- volatile uint16_t brake_max = 0

  Brake sensor max output voltage (equals to BRAKE_UPPER_BOUND)
- volatile uint16_t brake_min = 4095

  Brake sensor min output voltage (equals to BRAKE_LOWER_BOUND)
- volatile int bufn

  DMA buffer pointer.
- volatile int obufn

  DMA buffer pointer.
- volatile uint16_t buf [4][128 ∗4]

  DMA buffers: BUFFERS number of buffers each of BUFFER_LENGTH size; DMA is configured in cyclic mode: after one of BUFFERS is filled then DMA transfer head moves to next buffer in circular indexing.

8.5.1 Detailed Description

8.5.2 Macro Definition Documentation

8.5.2.1 BUFFERS

```
#define BUFFERS 4
```

Number of DMA buffers.

Warning

Must be a power of two

Definition at line 34 of file model.cpp.

8.5.3 Function Documentation

8.5.3.1 ADC_Handler()

```
void ADC_Handler ( )
```

ADC IRQ handler. When ADC buffer is filled DMA pointer is linked to next buffer available. Then acquired data are filtered.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 356 of file model.cpp.

8.5.3.2 get_apps_plausibility()

```
volatile bool get_apps_plausibility ( )
```

This function returns the value of APPS plausibility retrieved by CAN servizi network, if online, or by analog signal.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Return values

true	APPS plausibility
false	APPS implausibility

Definition at line 439 of file model.cpp.

8.5.3.3 get_brake_percentage()

```
volatile uint8_t get_brake_percentage ( )
```

This function returns the value of the brake pedal position sensor in percentage, retrieved by CAN servizi network, if online, or by analog signal.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Returns

Brake pedal position sensor percentage value

Definition at line 435 of file model.cpp.

8.5.3.4 get_brake_plausibility()

```
volatile bool get_brake_plausibility ( )
```

This function returns the value of brake plausibility retrieved by CAN servizi network, if online, or by analog signal.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Return values

true	Brake plausibility
false	Brake implausibility

Definition at line 443 of file model.cpp.

8.5.3.5 get_SC_value()

```
volatile uint16_t get_SC_value ( )
```

This function returns the value of the SC.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Returns

SC value

Definition at line 447 of file model.cpp.

8.5.3.6 get_tps1_percentage()

```
volatile uint8_t get_tps1_percentage ( )
```

This function returns the value of the first APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Returns

First APPS percentage value

Definition at line 427 of file model.cpp.

8.5.3.7 get_tps2_percentage()

```
volatile uint8_t get_tps2_percentage ( )
```

This function returns the value of the second APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Returns

Second APPS percentage value

Definition at line 431 of file model.cpp.

8.5.3.8 model_init()

```
void model_init ( )
```

This function initializes hardware board. Both APPS and brake pedal position sensor are acquired by analog signals for fault tolerance in case of CAN servizi fault.

This function initializes hardware board.

ADC peripheral is initialized with ADC_FREQ_MAX clock and with 12bits of resolution.

ADC peripheral is then configured in free running mode for continuous acquisitions.

ADC channels are enabled according to ADC_CHANNELS_LIST.

ADN End of Receive Buffer Interrupt is enabled for trigger interrupt when

Then digital GPIO ports for AIR+, AIR-, PRE, BUZZER, AIRbutton, RTDbutton are enabled.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)
Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 391 of file model.cpp.

8.6 RTDS

Functions

- void ready_to_drive_sound_start ()
  This function starts RTDS through a buzzer.

8.6.1 Detailed Description

8.6.2 Function Documentation

8.6.2.1 ready_to_drive_sound_start()

```
void ready_to_drive_sound_start ( )
```

This function starts RTDS through a buzzer.

This function starts RTDS.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 26 of file rtds.cpp.

8.7 Finite State Machine (FSM)

Macros

- #define SC_THRES 600

  SC voltage minimum value.
- #define RunBK 10

  Brake pedal position percentage threshold to pass from NOTDRIVE to DRIVE.
- #define RTDBK 10

  Brake pedal position percentage threshold to pass from HVON to DRIVE.
- #define REGEN_BK_PERCENTAGE 40

  Brake pedal position percentage threshold to active regen request.

Typedefs

- typedef enum enum_nodeState e_nodeState

  FSM states enum.

Enumerations

- enum enum_nodeState {
  STAND = 0, HVON, DRIVE, NOTDRIVE,
  MAX_STATES }

  FSM states enum.

Functions

- e_nodeState getState ()

  Return current state on the FSM.
- void setState (e_nodeState newState)

  Set current state on the FSM.
- void stand ()

  STAND state task on the FSM: ignition of the car. If AIR button is pressed and SC voltage value is greater than SC_THRES then current state passes to HVON, activating Precharge, AIR- and AIR+.
- void hvon ()

  HV ON state task on the FSM: active high voltage.
- void drive ()

  DRIVE state task on the FSM.
- void notdrive ()

  NOT DRIVE state task on the FSM: inconsistent pedals values.
- void state_dispatch ()

  Execute current state task.

Variables

- volatile bool RTD = false

  Ready To Drive flag.
- volatile e_nodeState current_state = STAND

  Current state on the FSM.
- void(∗ state_table [ ])(void) = {stand, hvon, drive, notdrive}

  FSM tasks function pointers.

8.7.1 Detailed Description

8.7.2 Enumeration Type Documentation

8.7.2.1 enum_nodeState

`enum enum_nodeState`

FSM states enum.

Enumerator

STAND	STAND state
HVON	HV ON state
DRIVE	DRIVE state
NOTDRIVE	NOT DRIVE state
MAX_STATES	max number of states

Definition at line 38 of file states.h.

8.7.3 Function Documentation

8.7.3.1 drive()

`void drive ( )`

DRIVE state task on the FSM.

If pedals values are consistent (no implausibility) and SC voltage value is greater than SC_THRES then torque or regen (if vehicle speed is $> \ 5\,km/h$) request is sent to inverter; request to inverter is done through CAN funzionale if online or through analog signals.

If SC voltage value is lower than SC_THRES, (SC undervoltage error), then current state passes to STAND.

If there are pedals implausibility then current state passes to NOT DRIVE state.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 162 of file states.cpp.

8.7.3.2 getState()

```
e_nodeState getState ( )
```

Return current state on the FSM.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Returns

current state on the FSM

Definition at line 68 of file states.cpp.

8.7.3.3 hvon()

```
void hvon ( )
```

HV ON state task on the FSM: active high voltage.

If RTD button is pressed, SC voltage value is greater than SC_THRES and brake pedal position percentage is greater than RTDBK then current state passes to DRIVE, triggering RTDS.

If SC voltage value is lower than SC_THRES, (SC undervoltage error), then current state passes to STAND.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Definition at line 125 of file states.cpp.

8.7.3.4 notdrive()

```
void notdrive ( )
```

NOT DRIVE state task on the FSM: inconsistent pedals values.

If SC voltage value is lower than SC_THRES, (SC undervoltage error), then current state passes to STAND state.

If pedals values return consistent (i.e. plausibility check verified), then current state passes to DRIVE state.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Definition at line 211 of file states.cpp.

8.7.3.5 setState()

```
void setState (
            e_nodeState newState )
```

Set current state on the FSM.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Parameters

`in`	*newState*	New state transition

Definition at line 82 of file states.cpp.

8.7.3.6 stand()

```
void stand ( )
```

STAND state task on the FSM: ignition of the car. If AIR button is pressed and SC voltage value is greater than SC_THRES then current state passes to HVON, activating Precharge, AIR- and AIR+.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 95 of file states.cpp.

8.7.3.7 state_dispatch()

```
void state_dispatch ( )
```

Execute current state task.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 232 of file states.cpp.

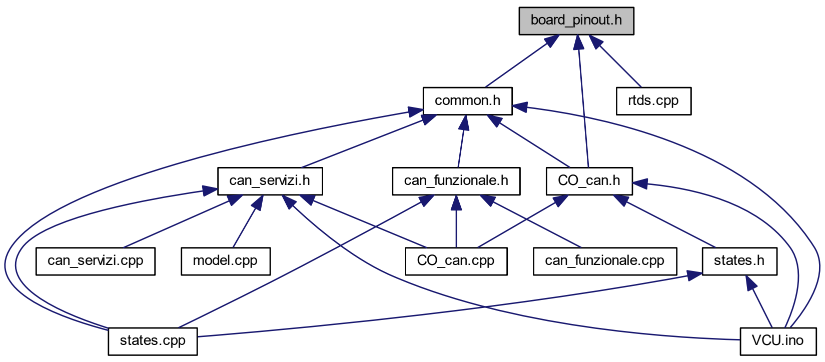
Chapter 9

File Documentation

9.1 board_pinout.h File Reference

Board pinout module header.

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



Macros

- #define CAN_FUNZIONALE Can1

  Pin on board dedicated to CAN funzionale port.
- #define CAN_SERVIZI Can0

  Pin on board dedicated to CAN servizi port.
- #define AIRcc 48

  Pin on board dedicated to AIR+.
- #define AIRGnd 49

  Pin on board dedicated to AIR-.
- #define PRE 47

  Pin on board dedicated to PRECHARGE.
- #define BUZZER 52

  Pin on board dedicated to buzzer for RTDS.

- #define AIRB 14

  Pin on board dedicated to AIR button.
- #define RTDB 15

  Pin on board dedicated to RTD button.
- #define FAN 9

  Pin on board dedicated to FAN.
- #define INVERTER_ANALOG_PIN DAC1

  Pin on board dedicated to inverter torque request analog signal.
- #define BRAKE_REGEN_PIN DAC0

  Pin on board dedicated to inverter regen request analog signal.
- #define TPS1_PIN A0

  Pin on board dedicated to first APPS (tps1)
- #define TPS1_ADC_CHAN_NUM ADC_CHER_CH7

  GPIO pin on the Atmel SAM3X8E processor corresponding to tps1 signal (AD7)
- #define TPS2_PIN A1

  Pin on board dedicated to second APPS (tps2)
- #define TPS2_ADC_CHAN_NUM ADC_CHER_CH6

  GPIO pin on the Atmel SAM3X8E processor corresponding to tps2 signal (AD6)
- #define BRAKE_PIN A2

  Pin on board dedicated to brake pedal position sensor.
- #define BRAKE_ADC_CHAN_NUM ADC_CHER_CH5

  GPIO pin on the Atmel SAM3X8E processor corresponding to brake signal (AD5)
- #define SC_PIN A3

  Pin on board dedicated to SC read signal.
- #define SC_ADC_CHAN_NUM ADC_CHER_CH4

  GPIO pin on the Atmel SAM3X8E processor corresponding to SC signal (AD4)

9.1.1 Detailed Description

Board pinout module header.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Date

2018

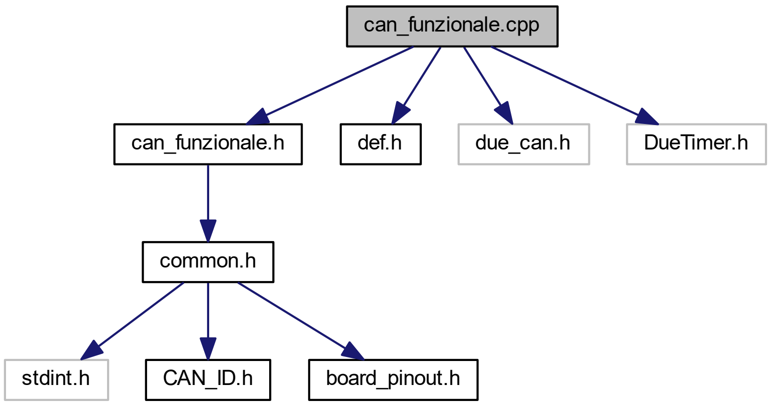
9.2 can_funzionale.cpp File Reference

CAN funzionale module implementation.

```
#include "can_funzionale.h"
#include "def.h"
#include <due_can.h>
```

```
#include <DueTimer.h>
```

Include dependency graph for can_funzionale.cpp:



Functions

- volatile bool can_funzionale_initialized ()

    This function returns CAN funzionale initialization status.
- void can_funzionale_send_sync ()

    This function sends a periodic CANOpen sync message to inverter slave node.
- void CAN_FUNZ_BOOTUP_CB (CAN_FRAME ∗frame)

    This function manage boot-up message sent over CAN funzionale network by inverter slave node. Upon boot-up message reception the VCU send a SDO client request for check inverter vendor ID; then inverter is considered online over CAN funzionale network.
- void CAN_FUNZ_VENDOR_ID_CB (CAN_FRAME ∗frame)

    This function manage SDO server response with inverter Vendor ID. VCU sends NMT operational and PDOs to enable PWM; then inverter is considered correctly configured and a timer is started for sending periodic sync messages.
- void CAN_FUNZ_GENERAL_CB (CAN_FRAME ∗frame)

    This function manage TPDO from inverter and deserializes data:
- bool can_funzionale_init ()

    This function initialize CAN funzionale hardware port with baudrate CAN_FUNZ_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from inverter slave node (filter = 0x00000700 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); mailbox 1 is configured for receiving vendorID SDO response from inverter (filter = 0x00000580 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); remaining mailboxes are configured for receiving TPDOs from inverter slave node (filter = 0x00000080, mask = 0x1FFFFCFF).
- volatile bool can_funzionale_online ()

    This function returns if inverter is online and active over CAN funzionale.
- void inverter_torque_request (uint16_t torque)

    This function send torque request to inverter. If inverter is active over CAN funzionale network then the request is done via RPDO1 viceversa it's done via analog signal.
- void inverter_regen_request (uint16_t regen)

    This function send regen request to inverter.
- volatile uint16_t get_torque_actual_value ()

    This function return the torque value requested by inverter to motor retrieved from TPDO1 from inverter over CAN funzionale network.

Variables

- volatile bool can_funz_initialized = false

  CAN funzionale initialization status flag (true if initialized)
- volatile bool inverter_online = false

  Inverter online status flag (true if online)
- volatile bool inverter_configured = false

  Inverter configured status flag (true if configured)
- volatile uint16_t torque_actual_value = 0

  Torque requested by inverter to motor.

9.2.1 Detailed Description

CAN funzionale module implementation.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

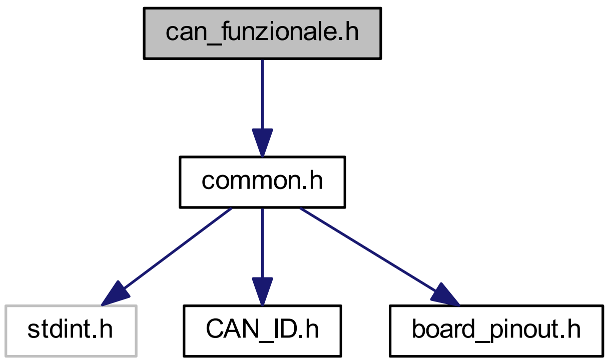
Date

2018

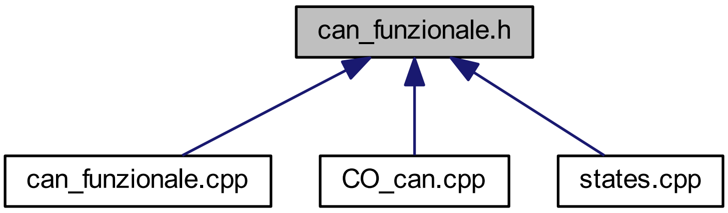
9.3 can_funzionale.h File Reference

CAN funzionale module header.

`#include "common.h"`
Include dependency graph for can_funzionale.h:



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



Functions

- bool can_funzionale_init ()

  This function initialize CAN funzionale hardware port with baudrate CAN_FUNZ_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from inverter slave node (filter = 0x00000700 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); mailbox 1 is configured for receiving vendorID SDO response from inverter (filter = 0x00000580 + INVERTER_NODE_ID, mask = 0x1FFFFFFF); remaining mailboxes are configured for receiving TPDOs from inverter slave node (filter = 0x00000080, mask = 0x1FFFFCFF).
- volatile bool can_funzionale_initialized ()

  This function returns CAN funzionale initialization status.
- volatile bool can_funzionale_online ()

  This function returns if inverter is online and active over CAN funzionale.
- void inverter_torque_request (uint16_t torque)

  This function send torque request to inverter. If inverter is active over CAN funzionale network then the request is done via RPDO1 viceversa it's done via analog signal.
- void inverter_regen_request (uint16_t regen)

  This function send regen request to inverter.
- volatile uint16_t get_torque_actual_value ()

  This function return the torque value requested by inverter to motor retrieved from TPDO1 from inverter over CAN funzionale network.

9.3.1 Detailed Description

CAN funzionale module header.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

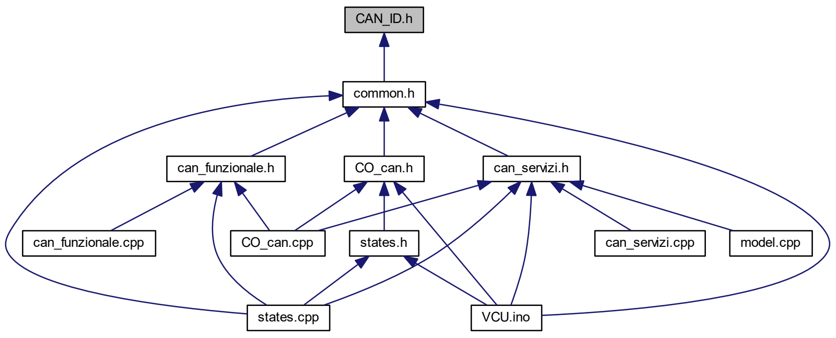
Date

2018

9.4 CAN_ID.h File Reference

CAN nodes ID definitions module.

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



Macros

- #define VCU_NODE_ID 2

    VCU Node ID.
- #define INVERTER_NODE_ID 1

    Inverter Node ID.
- #define SCU_FRONTAL_NODE_ID 1

    Frontal SCU Node ID.
- #define TCU_NODE_ID 4

    TCU Node ID.

9.4.1 Detailed Description

CAN nodes ID definitions module.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

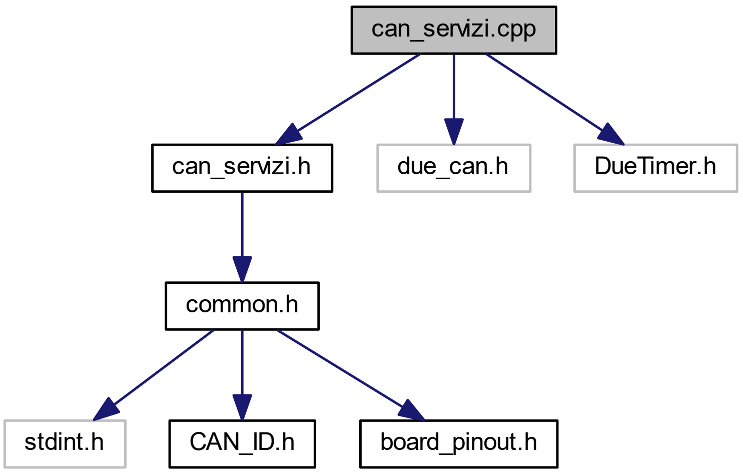
2018

9.5 can_servizi.cpp File Reference

CAN servizi module implementation.

```
#include "can_servizi.h"
#include <due_can.h>
#include <DueTimer.h>
```

Include dependency graph for can_servizi.cpp:



Functions

- void timeout ()

  This function is executed periodically after CAN servizi 'go Operational' NMT request is sent. When timeout occurs if next_pedals_seq_num is greater than curr_pedals_seq_num then frontal SCU is considered active, viceversa it is considered offline.
- volatile bool can_servizi_initialized ()

  This function returns CAN servizi initialization status.
- void CAN_SERV_BOOTUP_CB (CAN_FRAME ∗frame)

  This function manage boot-up messages sent over CAN servizi network by slave nodes.
- void CAN_SERV_GENERAL_CB (CAN_FRAME ∗frame)

  This function manage PDOs received over CAN servizi network and deserializes data:
- bool can_servizi_init ()

  This function initialize CAN servizi hardware port with baudrate CAN_SERV_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from CAN servizi slave nodes (filter = 0x00000700, mask = 0x1FFFFF80); remaining mailboxes are configured for receiving TPDOs from CAN servizi slave nodes (filter = 0x00000080, mask = 0x1FFF←FC80).
- void can_servizi_go_operational ()

  This function send a CANOpen master NMT message for request 'go to Operational' state to CAN servizi slave nodes (SCUs and TCU).
- volatile bool can_servizi_online ()

  This function returns if CAN servizi network is online.
- volatile bool tcs_online ()

This function returns if TCU node is active and online on the CAN servizi network.

- volatile uint8_t get_servizi_tps1 ()

    This function returns the value of the first APPS in percentage, retrieved by frontal SCU node over CAN servizi network.

- volatile uint8_t get_servizi_tps2 ()

    This function returns the value of the second APPS in percentage, retrieved by frontal SCU node over CAN servizi network.

- volatile uint8_t get_servizi_brake ()

    This function returns the value of brake pedal position sensor in percentage, retrieved by frontal SCU node over CAN servizi network.

- volatile bool get_servizi_apps_plausibility ()

    This function returns the value of APPS plausibility retrieved by frontal SCU node over CAN servizi network.

- volatile bool get_servizi_brake_plausibility ()

    This function returns the value of brake plausibility retrieved by frontal SCU node over CAN servizi network.

- volatile uint8_t get_tcs_torque_coefficient ()

    This function returns the value of torque limiter percentage retrieved by TCU node over CAN servizi network.

Variables

- volatile bool can_serv_initialized = false

    CAN servizi initialization status flag (true if initialized)

- volatile bool SCU_F_online = false

    Frontal SCU online status flag (true if online)

- volatile bool TCS_online = false

    TCS online status flag (true if online)

- volatile uint32_t curr_pedals_seq_num = 0

    Frontal SCU PDOtx1 current sequence number.

- volatile uint32_t next_pedals_seq_num = 0

    Frontal SCU PDOtx1 next sequence number.

- volatile uint8_t tps1_percentage = 0

    First APPS percentage value retrieved by frontal SCU node.

- volatile uint8_t tps2_percentage = 0

    Second APPS percentage value retrieved by frontal SCU node.

- volatile uint8_t brake_percentage = 0

    Brake pedal position sensor percentage value retrieved by frontal SCU node.

- volatile bool apps_plausibility = true

    APPS plausibility status retrieved by frontal SCU node.

- volatile bool brake_plausibility = true

    Brake plausibility status retrieved by frontal SCU node.

- volatile uint8_t tcs_coefficient = 0

    torque limiter percentage retrieved by TCU node

9.5.1 Detailed Description

CAN servizi module implementation.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

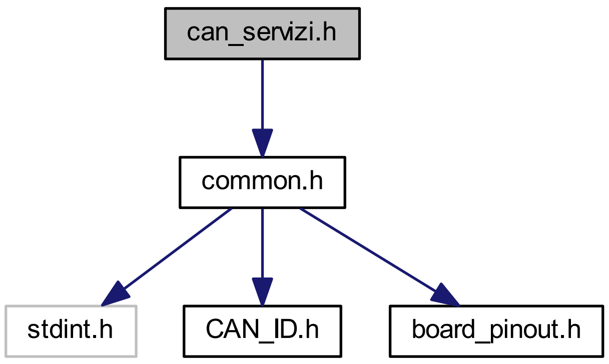
2018

9.6 can_servizi.h File Reference

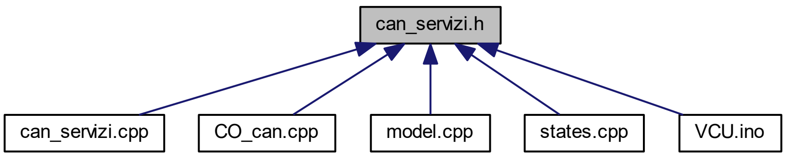
CAN servizi module header.

```
#include "common.h"
```

Include dependency graph for can_servizi.h:



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



Functions

- bool can_servizi_init ()

  This function initialize CAN servizi hardware port with baudrate CAN_SERV_BAUDRATE. Mailbox 0 is configured for receiving boot-up messages from CAN servizi slave nodes (filter = 0x00000700, mask = 0x1FFFFF80); remaining mailboxes are configured for receiving TPDOs from CAN servizi slave nodes (filter = 0x00000080, mask = 0x1FFF↩ FC80).
- volatile bool can_servizi_initialized ()

  This function returns CAN servizi initialization status.
- void can_servizi_go_operational ()

  This function send a CANOpen master NMT message for request 'go to Operational' state to CAN servizi slave nodes (SCUs and TCU).
- volatile bool can_servizi_online ()

  This function returns if CAN servizi network is online.

- volatile bool tcs_online ()

    This function returns if TCU node is active and online on the CAN servizi network.

- volatile uint8_t get_servizi_tps1 ()

    This function returns the value of the first APPS in percentage, retrieved by frontal SCU node over CAN servizi network.

- volatile uint8_t get_servizi_tps2 ()

    This function returns the value of the second APPS in percentage, retrieved by frontal SCU node over CAN servizi network.

- volatile uint8_t get_servizi_brake ()

    This function returns the value of brake pedal position sensor in percentage, retrieved by frontal SCU node over CAN servizi network.

- volatile bool get_servizi_apps_plausibility ()

    This function returns the value of APPS plausibility retrieved by frontal SCU node over CAN servizi network.

- volatile bool get_servizi_brake_plausibility ()

    This function returns the value of brake plausibility retrieved by frontal SCU node over CAN servizi network.

- volatile uint8_t get_tcs_torque_coefficient ()

    This function returns the value of torque limiter percentage retrieved by TCU node over CAN servizi network.

9.6.1 Detailed Description

CAN servizi module header.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

2018

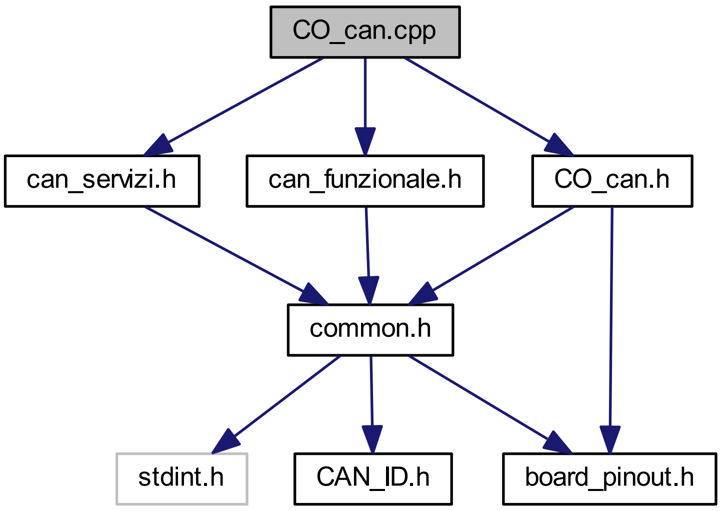
9.7 CO_can.cpp File Reference

CAN setup module implementation.

```
#include "CO_can.h"
#include "can_servizi.h"
```

`#include "can_funzionale.h"`

Include dependency graph for CO_can.cpp:



Functions

- bool can_init ()

    This function initializes both CAN funzionale and CAN servizi networks.

9.7.1 Detailed Description

CAN setup module implementation.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

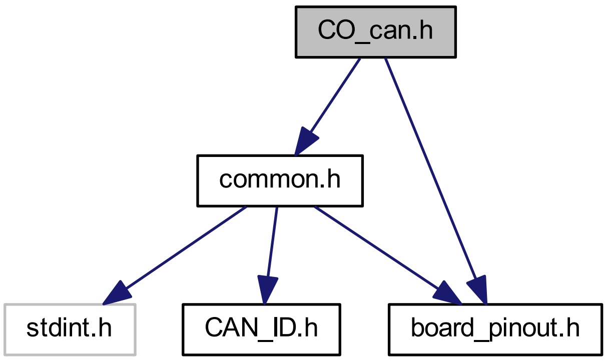
2018

9.8 CO_can.h File Reference

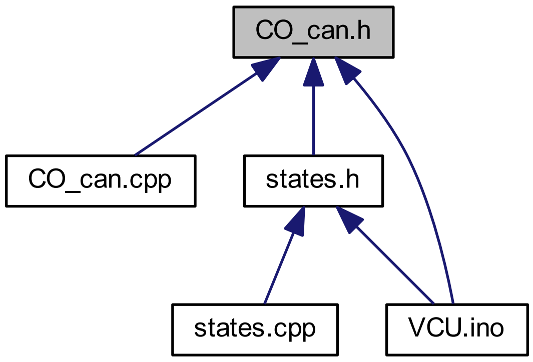
CAN setup header module.

```
#include "common.h"
#include "board_pinout.h"
```

Include dependency graph for CO_can.h:



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



Functions

- bool can_init ()

  This function initializes both CAN funzionale and CAN servizi networks.

9.8.1 Detailed Description

CAN setup header module.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Date

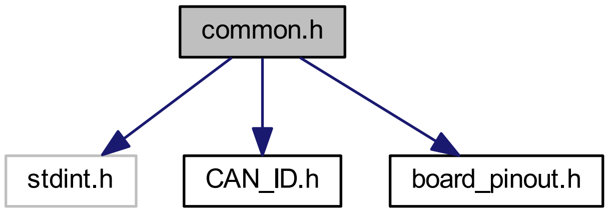
2018

9.9 common.h File Reference

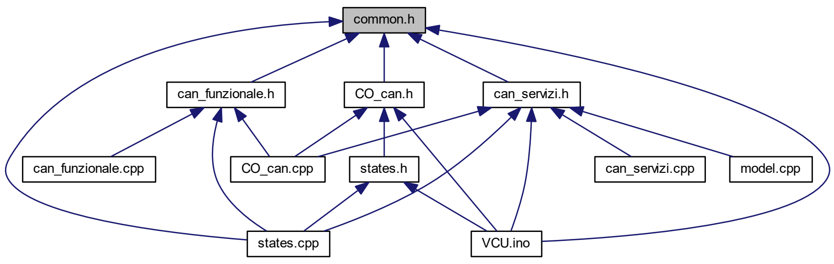
common macro definitions module

```
#include <stdint.h>
#include "CAN_ID.h"
#include "board_pinout.h"
```

Include dependency graph for common.h:



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



Macros

- #define CAN_FUNZ_BAUDRATE 1000000

    Defines CAN funzionale baudrate.

- #define CAN_SERV_BAUDRATE 1000000

Defines CAN servizi baudrate.

- #define SERIAL_BAUDRATE 115200

  Defines serial baudrate.
- #define INVERTER_TORQUE_MIN 0

  Defines inverter torque request lower bound.
- #define INVERTER_TORQUE_MAX 32767

  Defines inverter torque request upper bound.
- #define CAN_FUNZ_SYNC_PERIOD 5000

  Defines CAN funzionale sync message trasmission period.
- #define CAN_SERVIZI_TIMEOUT_PERIOD 30000

  Defines CAN servizi timeout period for fault check.

9.9.1 Detailed Description

common macro definitions module

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

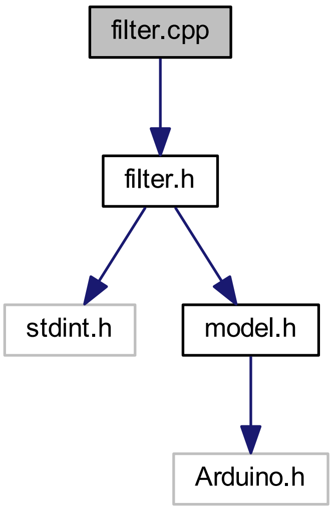
Date

2018

9.10 filter.cpp File Reference

Filter module implementation file.

`#include "filter.h"`
Include dependency graph for filter.cpp:



Macros

- #define USE_LOOP_UNROLLING (1)

  Flag macro for using or not loop unrolling into filter function.
- #define pos(x, offset) ((x) ∗ offset)

  Buffer indexing macro.

Functions

- uint16_t filter_buffer (volatile uint16_t ∗buffer, int size, unsigned offset)

  This function filters the input buffer with an average filter.

9.10.1 Detailed Description

Filter module implementation file.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

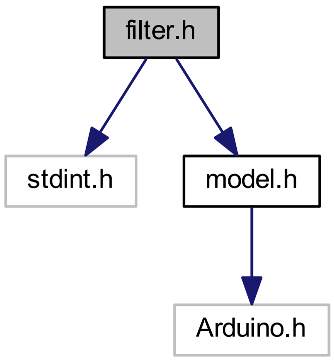
Date

2018

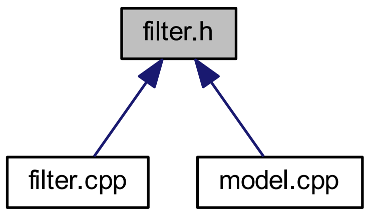
9.11 filter.h File Reference

Filter module header file.

```
#include <stdint.h>
#include "model.h"
```
Include dependency graph for filter.h:



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



Functions

- uint16_t filter_buffer (volatile uint16_t ∗buffer, int size, unsigned offset)

  This function filters the input buffer with an average filter.

9.11.1 Detailed Description

Filter module header file.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Date

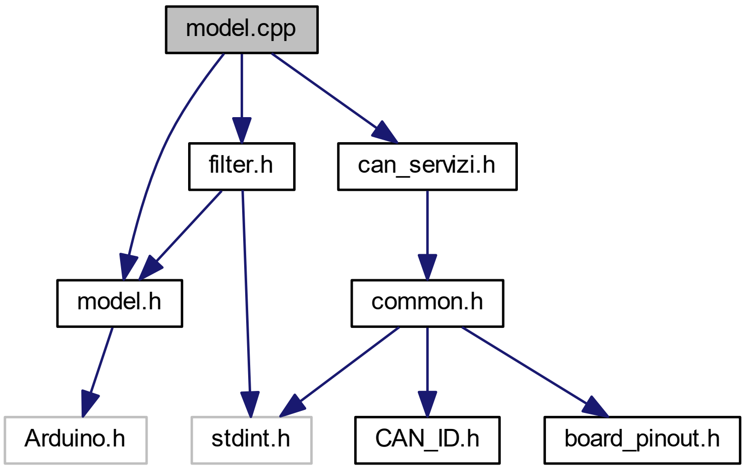
2018

9.12 model.cpp File Reference

Board model implementation file.

```
#include "model.h"
#include "filter.h"
```

```
#include "can_servizi.h"
```
Include dependency graph for model.cpp:



Macros

- #define ADC_BUFFER_SIZE 128

  Size (bytes) of buffer for store each ADC channel data with DMA.
- #define BUFFERS 4

  Number of DMA buffers.
- #define ADC_MIN 0

  ADC lower bound value.
- #define ADC_MAX 4095

  ADC upper bound value.
- #define ADC_CHANNELS_LIST TPS1_ADC_CHAN_NUM | TPS2_ADC_CHAN_NUM | BRAKE_ADC_CHAN_NUM | SC_ADC_CHAN_NUM

  List of ADC channels dedicated to each IO port.
- #define ADC_CHANNELS 4

  Number of ADC channels.
- #define TPS1_ADC_OFFSET 0

  Offset from DMA buffer.
- #define TPS2_ADC_OFFSET 1

  Offset from DMA buffer.
- #define BRAKE_ADC_OFFSET 2

  Offset from DMA buffer.
- #define SC_ADC_OFFSET 3

  Offset from DMA buffer.
- #define BUFFER_LENGTH ADC_BUFFER_SIZE ∗ ADC_CHANNELS

  Length, in bytes, of each DMA buffer.
- #define APPS_PLAUS_RANGE 10

  Size (bytes) of each ADC buffer.
- #define TPS1_UPPER_BOUND 2482

First APPS max output voltage (2V)
- #define TPS1_LOWER_BOUND 993

  First APPS min output voltage (0.8V)
- #define TPS2_UPPER_BOUND 1241

  Second APPS max output voltage (1V)
- #define TPS2_LOWER_BOUND 497

  Second APPS min output voltage (0.4V)
- #define BRAKE_UPPER_BOUND 0

  Brake sensor max output voltage (TODO: check Voutmax)
- #define BRAKE_LOWER_BOUND ADC_MAX

  Brake sensor min output voltage (TODO: check Voutmin)

Functions

- void ADC_Handler ()

  ADC IRQ handler. When ADC buffer is filled DMA pointer is linked to next buffer available. Then acquired data are filtered.
- void model_init ()

  This function initializes hardware board. Both APPS and brake pedal position sensor are acquired by analog signals for fault tolerance in case of CAN servizi fault.
- volatile uint8_t get_tps1_percentage ()

  This function returns the value of the first APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint8_t get_tps2_percentage ()

  This function returns the value of the second APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint8_t get_brake_percentage ()

  This function returns the value of the brake pedal position sensor in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile bool get_apps_plausibility ()

  This function returns the value of APPS plausibility retrieved by CAN servizi network, if online, or by analog signal.
- volatile bool get_brake_plausibility ()

  This function returns the value of brake plausibility retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint16_t get_SC_value ()

  This function returns the value of the SC.

Variables

- volatile uint8_t tps1_adc_percentage = 0

  First APPS percentage value retrieved by analog tps1 signal (TPS1_PIN)
- volatile uint8_t tps2_adc_percentage = 0

  Second APPS percentage value retrieved by analog tps2 signal (TPS2_PIN)
- volatile uint8_t brake_adc_percentage = 0

  Brake pedal position sensor percentage value retrieved by analog brake signal (BRAKE_PIN)
- volatile bool apps_adc_plausibility = true

  APPS plausibility status retrieved by analog acquisition.
- volatile bool brake_adc_plausibility = true

  Brake plausibility status retrieved by analog acquisition.
- volatile uint16_t tps1_value = 0

  First APPS value retrieved directly by analog tps1 signal (TPS1_PIN) and filtered after DMA buffer is filled entirely.

- volatile uint16_t tps2_value = 0

  Second APPS value retrieved directly by analog tps2 signal (TPS2_PIN) and filtered after DMA buffer is filled entirely.
- volatile uint16_t brake_value = 0

  Brake pedal position sensor value retrieved directly by analog brake signal (BRAKE_PIN) and filtered after DMA buffer is filled entirely.
- volatile uint16_t SC_value = 0

  SC value retrieved directly by analog SC signal (SC_PIN) and filtered after DMA buffer is filled entirely.
- volatile uint16_t tps1_max = 2482

  First APPS max output voltage (equals to TPS1_UPPER_BOUND)
- volatile uint16_t tps1_min = 993

  First APPS min output voltage (equals to TPS1_LOWER_BOUND)
- volatile uint16_t tps2_max = 1241

  Second APPS max output voltage (equals to TPS2_UPPER_BOUND)
- volatile uint16_t tps2_min = 497

  Second APPS min output voltage (equals to TPS2_LOWER_BOUND)
- volatile uint16_t brake_max = 0

  Brake sensor max output voltage (equals to BRAKE_UPPER_BOUND)
- volatile uint16_t brake_min = 4095

  Brake sensor min output voltage (equals to BRAKE_LOWER_BOUND)
- volatile int bufn

  DMA buffer pointer.
- volatile int obufn

  DMA buffer pointer.
- volatile uint16_t buf [4][128 *4]

  DMA buffers: BUFFERS number of buffers each of BUFFER_LENGTH size; DMA is configured in cyclic mode: after one of BUFFERS is filled then DMA transfer head moves to next buffer in circular indexing.

9.12.1 Detailed Description

Board model implementation file.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

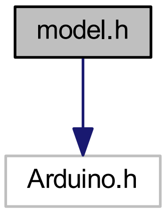
2018

9.13 model.h File Reference

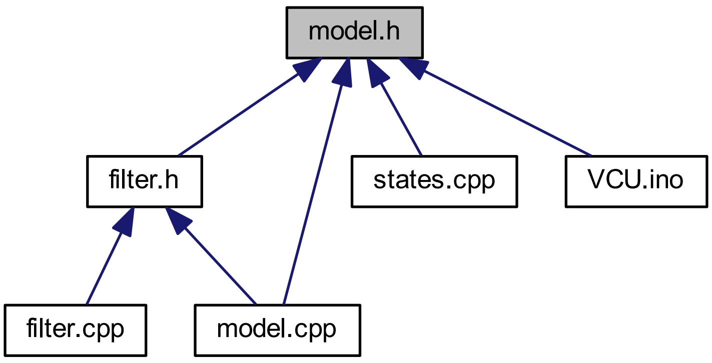
Board model header file.

```
#include <Arduino.h>
```

Include dependency graph for model.h:



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



Functions

- void model_init ()

  This function initializes hardware board. Both APPS and brake pedal position sensor are acquired by analog signals for fault tolerance in case of CAN servizi fault.
- volatile uint8_t get_tps1_percentage ()

  This function returns the value of the first APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint8_t get_tps2_percentage ()

  This function returns the value of the second APPS in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint8_t get_brake_percentage ()

  This function returns the value of the brake pedal position sensor in percentage, retrieved by CAN servizi network, if online, or by analog signal.
- volatile bool get_apps_plausibility ()

This function returns the value of APPS plausibility retrieved by CAN servizi network, if online, or by analog signal.

- volatile bool get_brake_plausibility ()

  This function returns the value of brake plausibility retrieved by CAN servizi network, if online, or by analog signal.
- volatile uint16_t get_SC_value ()

  This function returns the value of the SC.

9.13.1 Detailed Description

Board model header file.

Author

Arella Matteo
(mail: arella.1646983@studenti.uniroma1.it)

Date

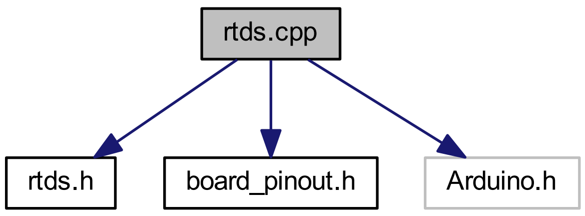
2018

9.14 rtds.cpp File Reference

RTDS module implementation file.

```
#include "rtds.h"
#include "board_pinout.h"
#include <Arduino.h>
```

Include dependency graph for rtds.cpp:



Functions

- void ready_to_drive_sound_start ()

  This function starts RTDS through a buzzer.

9.14.1 Detailed Description

RTDS module implementation file.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

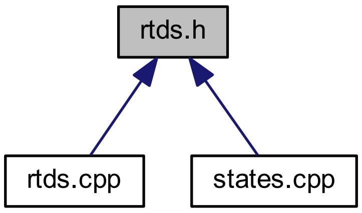
Date

2018

9.15 rtds.h File Reference

RTDS header file.

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



Functions

- void ready_to_drive_sound_start ()
  This function starts RTDS through a buzzer.

9.15.1 Detailed Description

RTDS header file.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

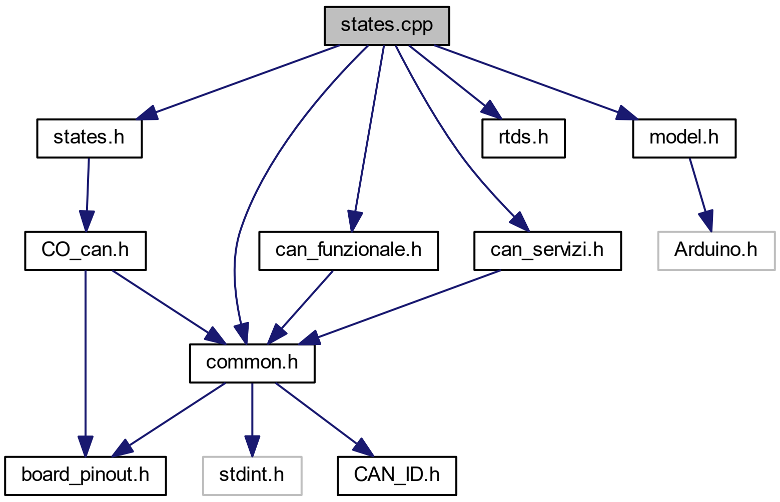
2018

9.16 states.cpp File Reference

FSM implementation file.

```
#include "states.h"
#include "can_funzionale.h"
#include "can_servizi.h"
#include "rtds.h"
#include "model.h"
#include "common.h"
```

Include dependency graph for states.cpp:



Macros

- #define SC_THRES 600

  SC voltage minimum value.
- #define RunBK 10

  Brake pedal position percentage threshold to pass from NOTDRIVE to DRIVE.
- #define RTDBK 10

  Brake pedal position percentage threshold to pass from HVON to DRIVE.
- #define REGEN_BK_PERCENTAGE 40

  Brake pedal position percentage threshold to active regen request.

Functions

- e_nodeState getState ()

  Return current state on the FSM.
- void setState (e_nodeState newState)

  Set current state on the FSM.
- void stand ()

STAND state task on the FSM: ignition of the car. If AIR button is pressed and SC voltage value is greater than SC_THRES then current state passes to HVON, activating Precharge, AIR- and AIR+.

- void hvon ()

    HV ON state task on the FSM: active high voltage.

- void drive ()

    DRIVE state task on the FSM.

- void notdrive ()

    NOT DRIVE state task on the FSM: inconsistent pedals values.

- void state_dispatch ()

    Execute current state task.

Variables

- volatile bool RTD = false

    Ready To Drive flag.

- volatile e_nodeState current_state = STAND

    Current state on the FSM.

- void(∗ state_table [ ])(void) = {stand, hvon, drive, notdrive}

    FSM tasks function pointers.

9.16.1 Detailed Description

FSM implementation file.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

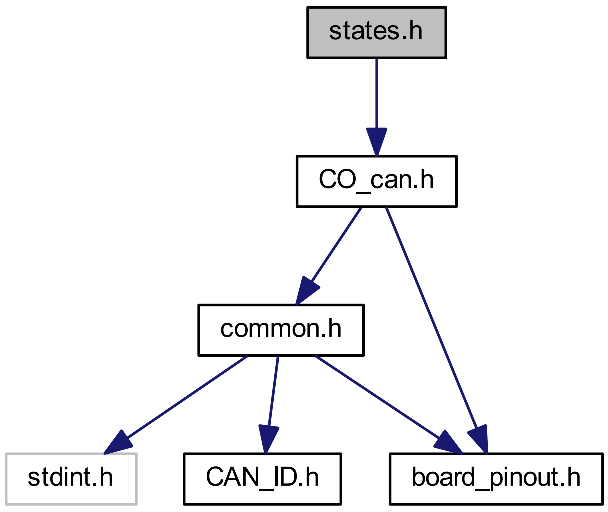
Date

2018

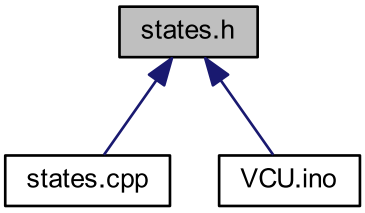
9.17 states.h File Reference

FSM header file.

```
#include "CO_can.h"
```
Include dependency graph for states.h:



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



Typedefs

- typedef enum enum_nodeState e_nodeState

  FSM states enum.

Enumerations

- enum enum_nodeState {
  STAND = 0, HVON, DRIVE, NOTDRIVE,
  MAX_STATES }

  FSM states enum.

Functions

- e_nodeState getState ()

    Return current state on the FSM.

- void setState (e_nodeState newState)

    Set current state on the FSM.

- void state_dispatch ()

    Execute current state task.

9.17.1 Detailed Description

FSM header file.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

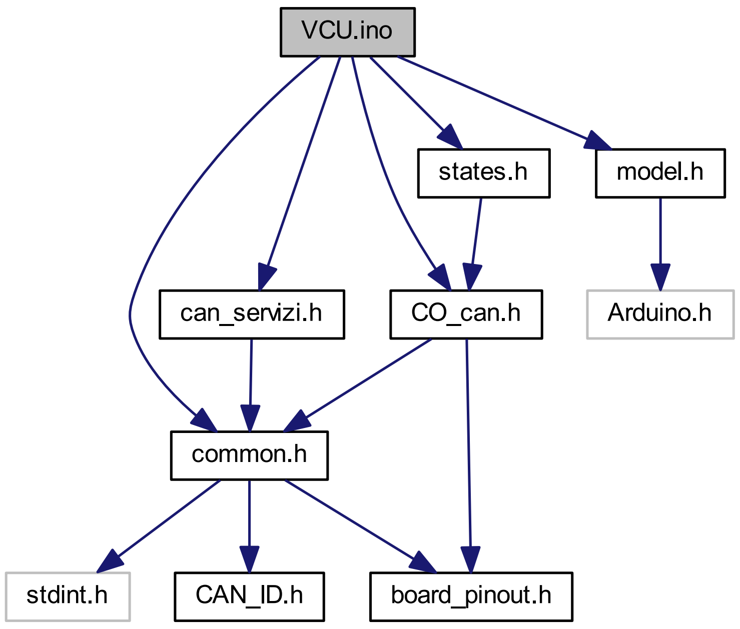
2018

9.18 VCU.ino File Reference

Main module file.

```
#include "common.h"
#include "CO_can.h"
#include "can_servizi.h"
#include "model.h"
#include "states.h"
```

Include dependency graph for VCU.ino:



Functions

- void setup ()

    This function perform basic board setup.

- void loop ()

    This function is called into endless while main loop. It takes care of dispatching states of the finite state machine (TODO: see states)

9.18.1 Detailed Description

Main module file.

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Date

2018

9.18.2 Function Documentation

9.18.2.1 loop()

```
void loop ( )
```

This function is called into endless while main loop. It takes care of dispatching states of the finite state machine (TODO: see states)

Author

Arella Matteo
(mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 59 of file VCU.ino.

9.18.2.2 setup()

```
void setup ( )
```

This function perform basic board setup.

- It starts initializing both CAN funzionale (with inverter) and CAN servizi (with the two SCUs and TCS); if the comunication between inverter and VCU can't be established via CAN bus then the VCU is configured to request torque value to inverter by analog signal.
- It initializes board hardware (Board model)
- If the configuration over CAN servizi with the frontal SCU was successful then VCU (master) send an NMT request to go in 'Operational' state.

  Author

  Arella Matteo
  (mail: `arella.1646983@studenti.uniroma1.it`)

Definition at line 39 of file VCU.ino.

Index