Sensors Control Unit (SCU)

Firmware Specifications

History

Version	Author	Date
1.0	Arella Matteo	2018

Contents

Chapter 1

FastChargeSAE SCU firmware

CAN network arises from the need to digitize all those signals necessary for the operation of the car.

Two Arduino Due prototyping boards have been adopted for signal digitalization: first one located at the front of the vehicle, reserved for the acquisition of pedals, frontal suspensions and frontal wheel groups, the second one placed an the rear of the vehicle, to acquire rear suspensions, rear wheels and accelerometers.

Sensor acquisition boards will now be named SCU (Sensors Control Unit) and SCU FRONTAL, SCU REAR respectively for SCU located at the front and at the back of the vehicle.

Each board performs mainly two actions:

- Sensor acquisition
- Data transmission over CAN servizi network and over radio (for real time telemetry)

A protocol layer above the data link layer (CAN protocol) is implemented inspired by the CANOpen communication protocol; each node is addressable at the network level using a specific and unique ID for every node.

Each SCU board can be represented by a finite state machine with the following statuses: Initialisation, Pre-operational, Operational, Stopped. During power-up each node is in the Initialization state. 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 make the various SCUs pass between the various Pre-operational, Operational and Stopped states.

Each SCU sends PDOs with sensor data in synchronous mode only if it is in the Operational state.

The firmware for each node is selectable during the precompilation of the code from the directives present in SCU firmware selection.

Chapter 2

CAN Servizi network

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 servizi network has an unique ID into that network, according to this table:

NODE	NODE-ID
$SCU_{Frontal}$	1
VCU	2
SCU_{Rear}	3
TCU	4

SCU slave on power up sequence

1. **send BOOT-UP message after initialisation state**

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

2. **wait NMT 'go Operational' from VCU master node**

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

3. **periodically send TPDOs with sensors' data** Each node starts a timer with TIME_SLOT_PERIOD period. In this way one slot (or more) of TIME_SLOT_PERIOD is assigned to each node for transmission, so as to reduce CAN bus load.

START TIMER	packet #1	packet #2	packet #3	packet #4
VCU	$SCU_{Frontal}$	$SCU_{Frontal}$	SCU_{Rear}	TCU
	TRANSMISSION PERIOD			

TPDO configuration

TX NODE	Data	Unit	Data Length	Data Offset	#CAN packet	ID	Total Length
$SCU_{Frontal}$	First APPS	$\%$	$1\ Byte$	$[0:7]$	#1	TPDO1 + NODE-ID	4
	Second APPS	$\%$	$1\ Byte$	$[8:15]$			
	Brake	$\%$	$1\ Byte$	$[16:23]$			
	APPS plausibility	$bool$	$4\ bit$	$[24:27]$			
	Brake plausibility	$bool$	$4\ bit$	$[28:31]$			
	Right phonic wheel	rpm	$2\ Bytes$	$[0:15]$	#2	TPDO2 + NODE-ID	6
	Left phonic wheel	rpm	$2\ Bytes$	$[16:31]$			
	Right suspension	mm	$1\ Byte$	$[32:39]$			
	Left suspension	mm	$1\ Byte$	$[40:47]$			
SCU_{Rear}	Accel. X	m/s^2	$1\ Byte$	$[0:7]$	#3	TPDO1 + NODE-ID	8
	Accel. Z	m/s^2	$1\ Byte$	$[8:15]$			
	Right suspension	mm	$1\ Byte$	$[16:23]$			
	Left suspension	mm	$1\ Byte$	$[24:31]$			
	Right phonic wheel	rpm	$2\ Bytes$	$[32:47]$			
	Left phonic wheel	rpm	$2\ Bytes$	$[48:63]$			
TCU	Torque limiter	$\%$	$1\ Byte$	$[0:7]$	#4	TPDO1 + NODE-ID	1

where TPDO1 = 0x180 and TPDO2 = 0x280

Chapter 3

Board model

SCUs are based on an Atmel SAM3X8E board with an ARM Cortex-M3 microprocessor.

Analog input signals managed from SCUs are:

NODE	Signal
$SCU_{Frontal}$	First APPS
	Second APPS
	Brake
	Right phonic wheel
	Left phonic wheel
	Right suspension
	Left suspension
SCU_{Rear}	Accel. X
	Accel. Z
	Right suspension
	Left suspension
	Right phonic wheel
	Left phonic wheel

Board pinout is described in Board module.

Output rpm are returned following this formula:

$$rpm = NORMALIZE\_RPM * \left(\frac{60}{COGS\_NUMBER * \Delta_{TIMESTAMP}}\right)$$

where $NORMALIZE\_RPM$ normalizes $TIMESTAMP$ in seconds.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

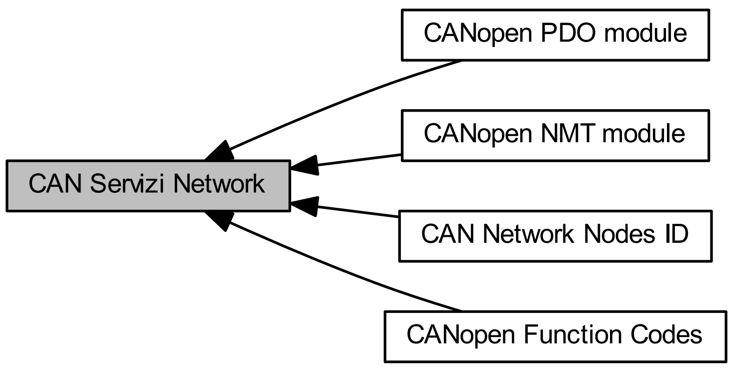
Here is a list of all documented files with brief descriptions:

Chapter 7

Module Documentation

7.1 CAN Servizi Network

Collaboration diagram for CAN Servizi Network:



Modules

- CAN Network Nodes ID
- CANopen Function Codes
- CANopen NMT module
- CANopen PDO module

Classes

- struct Message

    CANopen message struct.

Macros

- #define Message_Initializer {0,0,{0,0,0,0,0,0,0,0}}

    CANopen static message initializer.

Functions

- void canSend (Message ∗m)

    This function send a CANopen message over CAN servizi network.
- void **CAN_general_callback** (CAN_FRAME ∗frame)
- void initCAN ()

    This function initialize CAN/CANopen interfaces and communication.

7.1.1 Detailed Description

7.1.2 Function Documentation

7.1.2.1 canSend()

```
void canSend (
            Message ∗ m )
```

This function send a CANopen message over CAN servizi network.

Author

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

Parameters

in	*m*	CANopen message to send

Definition at line 20 of file CO_can.cpp.

7.1.2.2 initCAN()

```
void initCAN ( )
```

This function initialize CAN/CANopen interfaces and communication.

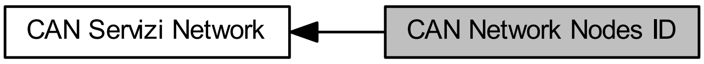
Author

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

Definition at line 42 of file CO_can.cpp.

7.2 CAN Network Nodes ID

Collaboration diagram for CAN Network Nodes ID:



Macros

- #define SCU_FRONTAL_NODE_ID 1

  Frontal SCU node ID on CAN servizi network.
- #define VCU_NODE_ID 2

  VCU node ID on CAN servizi network.
- #define SCU_REAR_NODE_ID 3

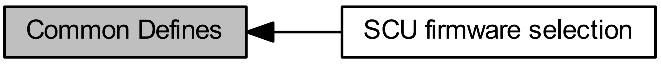
  Rear SCU node ID on CAN servizi network.
- #define TCU_NODE_ID 4

  TCU node ID on CAN servizi network.

7.2.1 Detailed Description

7.3 Common Defines

Collaboration diagram for Common Defines:



Modules

- SCU firmware selection

Macros

- #define CAN_BAUDRATE 1000000

  CAN network baud rate [bps].
- #define SERIAL_BAUDRATE 115200

  Serial UART baud rate [bps].
- #define TIME_SLOT_PERIOD 4000

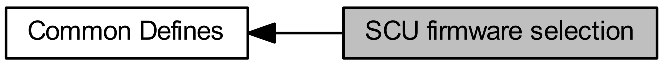
  Timer period [ms].
- #define TIMER Timer3

  CANopen timer.

7.3.1 Detailed Description

7.4 SCU firmware selection

Collaboration diagram for SCU firmware selection:



Macros

- #define _FRONTAL_

  Macro for selecting frontal SCU firmware.
- #define _RETRO_

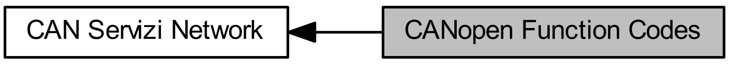
  Macro for selecting rear SCU firmware.

7.4.1 Detailed Description

The firmware for each node is selectable during the precompilation of the code from the directives present in that module. Comment/uncomment those macros for active/deactive selected SCU firmware.

7.5 CANopen Function Codes

Collaboration diagram for CANopen Function Codes:



Macros

- #define NMT 0x0

  NMT function code.
- #define SYNC 0x1

  SYNC function code.
- #define TIME_STAMP 0x2

  TIME_STAMP function code.
- #define PDO1tx 0x3

  PDO1tx function code.
- #define PDO1rx 0x4

  PDO1rx function code.
- #define PDO2tx 0x5

  PDO2tx function code.
- #define PDO2rx 0x6

  PDO2rx function code.
- #define PDO3tx 0x7

  PDO3tx function code.
- #define PDO3rx 0x8

  PDO3rx function code.
- #define PDO4tx 0x9

  PDO4tx function code.
- #define PDO4rx 0xA

  PDO4rx function code.
- #define SDOtx 0xB

  SDOtx function code.
- #define SDOrx 0xC

  SDOrx function code.
- #define NODE_GUARD 0xE

  NODE GUARD function code.
- #define LSS 0xF

  LSS function code.
- #define GET_FUNC_CODE(COB_ID) (COB_ID >> 7)

  Extract function code from COB-ID.
- #define SET_FUNC_CODE(COB_ID) (COB_ID << 7)

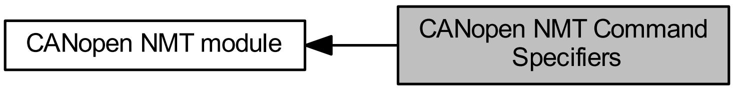
  Set function code to COB-ID.

7.5.1 Detailed Description

CANopen function codes defined in DS301 profile.

7.6 CANopen NMT Command Specifiers

Collaboration diagram for CANopen NMT Command Specifiers:



Macros

- #define NMT_Start_Node 0x01

  'go Operational' command specifier
- #define NMT_Stop_Node 0x02

  'stop Node' command specifier
- #define NMT_Enter_PreOperational 0x80

  'go PreOperational' command specifier
- #define NMT_Reset_Node 0x81

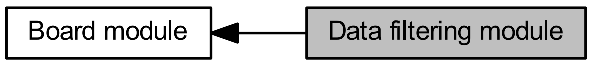
  'reset Node' command specifier
- #define NMT_Reset_Comunication 0x82

  'reset Communication' command specifier

7.6.1 Detailed Description

7.7 Data filtering module

Collaboration diagram for Data filtering module:



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.

7.7.1 Detailed Description

7.7.2 Function Documentation

7.7.2.1 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

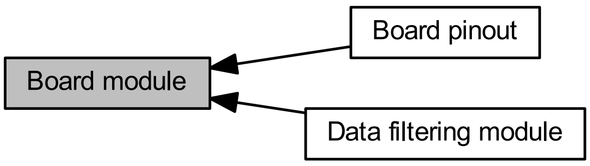
Returns

Filtered data

Definition at line 39 of file filter.cpp.

7.8 Board module

Collaboration diagram for Board module:



Modules

- Data filtering module
- Board pinout

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 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)
- #define SUSPENSIONS_MIN 0

  Minimum suspension stroke [mm].
- #define SUSPENSIONS_ADC_MAX ADC_MAX

  Maximum suspension sensor V_{OUT}.
- #define SUSP_STROKE_NORMALIZE (SUSP_STROKE_EXTENSION / SUSPENSIONS_ADC_MAX)

  Suspension stroke voltage normalizer.

- #define COGS_NUMBER 30.0d

    Number of phonic wheel's cogs.

- #define NORMALIZE_RPM 1000000.0d

    Normalize time domain [μs].

- #define RPM_MIN 10

    Rpm lower bound under that rpm are forced to zero.

- #define ACCELEROMETER_MAX_G 5.0d

    Accelerometer sensor maximum value [m/s^2].

- #define ACCELEROMETER_NORMALIZE 2.0d $*$ ACCELEROMETER_MAX_G / ADC_MAX

    Accelerometer sensor voltage normalizer.

- #define APPS_PLAUS_RANGE 10

    Maximum percentage deviation of pedal travel between two APPS.

- #define SCU_FRONTAL_ADC_CHANNELS 5

    Number of ADC channels used in frontal SCU board.

- #define SCU_FRONTAL_ADC_CHANNELS_LIST TPS1_ADC_CHAN_NUM | TPS2_ADC_CHAN_NUM | BRAKE_ADC_CHAN_NUM | FR_SX_ADC_CHAN_NUM | FR_DX_ADC_CHAN_NUM

    List of ADC channels dedicated to each IO port in frontal SCU board.

- #define SCU_RETRO_ADC_CHANNELS 4

    Number of ADC channels used in rear SCU board.

- #define SCU_RETRO_ADC_CHANNELS_LIST ACC_X_ADC_CHAN_NUM | ACC_Z_ADC_CHAN_NUM | RT_SX_ADC_CHAN_NUM | RT_DX_ADC_CHAN_NUM

    List of ADC channels dedicated to each IO port in retro SCU board.

- #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 FR_SX_ADC_OFFSET 3

    Offset from DMA buffer.

- #define FR_DX_ADC_OFFSET 4

    Offset from DMA buffer.

- #define ACC_X_ADC_OFFSET 0

    Offset from DMA buffer.

- #define ACC_Z_ADC_OFFSET 1

    Offset from DMA buffer.

- #define RT_SX_ADC_OFFSET 2

    Offset from DMA buffer.

- #define RT_DX_ADC_OFFSET 3

    Offset from DMA buffer.

- #define BUFFER_LENGTH ADC_BUFFER_SIZE $*$ ADC_CHANNELS

    Length, in bytes, of each DMA buffer.

- #define SUSP_STROKE_EXTENSION 75.0

    Maximum suspension stroke [mm].

Functions

- void fr_sx_pulse ()

  EXTI IRQ handler. External interrupt handler executed when frontal left wheel encoder finds a hole into phonic wheel.
- void fr_dx_pulse ()

  EXTI IRQ handler. External interrupt handler executed when frontal right wheel encoder finds a hole into phonic wheel.
- volatile uint16_t get_fr_sx_rpm ()

  If rpm value is lower than RPM_MIN, output is forced to zero.
- volatile uint16_t get_fr_dx_rpm ()

  If rpm value is lower than RPM_MIN, output is forced to zero.
- 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.
- volatile uint16_t get_rt_sx_rpm ()

  This function returns retro left wheel velocity [rpm].
- volatile uint16_t get_rt_dx_rpm ()

  This function returns retro right wheel velocity [rpm].

Variables

- volatile int bufn

  DMA buffer pointer.
- volatile int obufn

  DMA buffer pointer.
- volatile uint16_t buf [BUFFERS][BUFFER_LENGTH]

  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.
- 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 tps1_max = TPS1_UPPER_BOUND

  First APPS max output voltage (equals to TPS1_UPPER_BOUND)
- volatile uint16_t tps1_min = TPS1_LOWER_BOUND

  First APPS min output voltage (equals to TPS1_LOWER_BOUND)
- volatile uint16_t tps2_max = TPS2_UPPER_BOUND

  Second APPS max output voltage (equals to TPS2_UPPER_BOUND)
- volatile uint16_t tps2_min = TPS2_LOWER_BOUND

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

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

  Brake sensor min output voltage (equals to BRAKE_LOWER_BOUND)
- volatile uint8_t tps1_percentage = 0

  First APPS percentage value retrieved by tps1 signal (TPS1_PIN)

- volatile uint8_t tps2_percentage = 0

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

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

  APPS plausibility status.
- volatile bool brake_plausibility = true

  Brake plausibility status.
- volatile uint8_t fr_sx_susp

  Frontal left suspension stroke [mm].
- volatile uint8_t fr_dx_susp

  Frontal right suspension stroke [mm].
- volatile uint16_t fr_sx_rpm = 0

  Frontal left wheel velocity [rpm].
- volatile uint16_t fr_dx_rpm = 0

  Frontal right wheel velocity [rpm].
- volatile unsigned long fr_sx_prev

  Frontal left wheel encoder previous timestamp.
- volatile unsigned long fr_sx_curr

  Frontal left wheel encoder current timestamp.
- volatile unsigned long fr_dx_prev

  Frontal right wheel encoder previous timestamp.
- volatile unsigned long fr_dx_curr

  Frontal right wheel encoder current timestamp.
- volatile uint8_t tps1_percentage

  First APPS percentage value retrieved by tps1 signal (TPS1_PIN)
- volatile uint8_t tps2_percentage

  Second APPS percentage value retrieved by tps2 signal (TPS2_PIN)
- volatile uint8_t brake_percentage

  Brake pedal position sensor percentage value retrieved by brake signal (BRAKE_PIN)
- volatile bool apps_plausibility

  APPS plausibility status.
- volatile bool brake_plausibility

  Brake plausibility status.
- volatile uint8_t fr_sx_susp

  Frontal left suspension stroke [mm].
- volatile uint8_t fr_dx_susp

  Frontal right suspension stroke [mm].
- volatile uint8_t acc_x_value

  Accelerometer X value [m/s^2].
- volatile uint8_t acc_z_value

  Accelerometer Z value [m/s^2].
- volatile uint8_t rt_sx_susp

  Retro left suspension stroke [mm].
- volatile uint8_t rt_dx_susp

  Retro right suspension stroke [mm].

7.8.1 Detailed Description

7.8.2 Macro Definition Documentation

7.8.2.1 BUFFERS

```
#define BUFFERS 4
```

Number of DMA buffers.

Warning

Must be a power of two

Definition at line 30 of file model.cpp.

7.8.3 Function Documentation

7.8.3.1 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 675 of file model.cpp.

7.8.3.2 fr_dx_pulse()

```
void fr_dx_pulse ( )
```

EXTI IRQ handler. External interrupt handler executed when frontal right wheel encoder finds a hole into phonic wheel.

Author

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

Definition at line 491 of file model.cpp.

7.8.3.3 fr_sx_pulse()

```
void fr_sx_pulse ( )
```

EXTI IRQ handler. External interrupt handler executed when frontal left wheel encoder finds a hole into phonic wheel.

Author

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

Definition at line 478 of file model.cpp.

7.8.3.4 get_fr_dx_rpm()

```
volatile uint16_t get_fr_dx_rpm ( )
```

If rpm value is lower than RPM_MIN, output is forced to zero.

This function returns frontal right wheel velocity [rpm].

Author

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

Returns

Frontal right wheel rpm

Definition at line 520 of file model.cpp.

7.8.3.5 get_fr_sx_rpm()

```
volatile uint16_t get_fr_sx_rpm ( )
```

If rpm value is lower than RPM_MIN, output is forced to zero.

This function returns frontal left wheel velocity [rpm].

Author

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

Returns

Frontal left wheel rpm

Definition at line 505 of file model.cpp.

7.8.3.6 get_rt_dx_rpm()

```
volatile uint16_t get_rt_dx_rpm ( )
```

This function returns retro right wheel velocity [rpm].

Author

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

Returns

Retro right wheel rpm

7.8.3.7 get_rt_sx_rpm()

```
volatile uint16_t get_rt_sx_rpm ( )
```

This function returns retro left wheel velocity [rpm].

Author

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

Returns

Retro left wheel rpm

7.8.3.8 model_init()

```
void model_init ( )
```

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 SCU_FRONTAL_ADC_CHANNELS_LIST or SCU_RETRO_ADC_CHANNELS_LIST.

ADC End of Receive Buffer Interrupt is enabled for triggering interrupt when DMA has filled entire buffer.

Then EXTI are enabled for triggering interrupt by wheel encoders.

Author

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

Definition at line 708 of file model.cpp.

7.8.4 Variable Documentation

7.8.4.1 apps_plausibility [1/2]

```
volatile bool apps_plausibility
```

APPS plausibility status.

APPS plausibility flag.

Definition at line 338 of file model.cpp.

7.8.4.2 apps_plausibility [2/2]

```
volatile bool apps_plausibility = true
```

APPS plausibility status.

APPS plausibility flag.

Definition at line 338 of file model.cpp.

7.8.4.3 brake_percentage [1/2]

```
volatile uint8_t brake_percentage
```

Brake pedal position sensor percentage value retrieved by brake signal (BRAKE_PIN)

Brake pedal position sensor percentage value.

Definition at line 332 of file model.cpp.

7.8.4.4 brake_percentage [2/2]

```
volatile uint8_t brake_percentage = 0
```

Brake pedal position sensor percentage value retrieved by brake signal (BRAKE_PIN)

Brake pedal position sensor percentage value.

Definition at line 332 of file model.cpp.

7.8.4.5 brake_plausibility [1/2]

```
volatile bool brake_plausibility
```

Brake plausibility status.

Brake plausibility flag.

Definition at line 344 of file model.cpp.

7.8.4.6 brake_plausibility [2/2]

```
volatile uint8_t brake_plausibility = true
```

Brake plausibility status.

Brake plausibility flag.

Definition at line 344 of file model.cpp.

7.8.4.7 tps1_percentage [1/2]

```
volatile uint8_t tps1_percentage
```

First APPS percentage value retrieved by tps1 signal (TPS1_PIN)

First APPS percentage value.

Definition at line 319 of file model.cpp.

7.8.4.8 tps1_percentage [2/2]

```
volatile uint8_t tps1_percentage = 0
```

First APPS percentage value retrieved by tps1 signal (TPS1_PIN)

First APPS percentage value.

Definition at line 319 of file model.cpp.

7.8.4.9 tps2_percentage [1/2]

```
volatile uint8_t tps2_percentage
```

Second APPS percentage value retrieved by tps2 signal (TPS2_PIN)

Second APPS percentage value.

Definition at line 325 of file model.cpp.

7.8.4.10 tps2_percentage [2/2]

```
volatile uint8_t tps2_percentage = 0
```

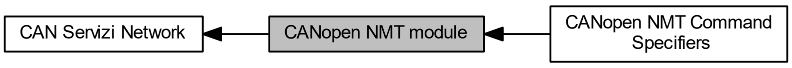
Second APPS percentage value retrieved by tps2 signal (TPS2_PIN)

Second APPS percentage value.

Definition at line 325 of file model.cpp.

7.9 CANopen NMT module

Collaboration diagram for CANopen NMT module:



Modules

- CANopen NMT Command Specifiers

Functions

- void proceedNMTstateChange (Message ∗m)

  According to CANopen NMT Command Specifiers, upon NMT reception from VCU master node, SCU change current state.
- void slaveSendBootUp ()

  This function sends a slave boot-up message over CAN servizi network.

7.9.1 Detailed Description

7.9.2 Function Documentation

7.9.2.1 proceedNMTstateChange()

```
void proceedNMTstateChange (
            Message * m )
```

According to CANopen NMT Command Specifiers, upon NMT reception from VCU master node, SCU change current state.

This function manages an NMT request from master node on CAN servizi network.

Author

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

Parameters

in	*m*	Received NMT message

Definition at line 26 of file nmt.cpp.

7.9.2.2 slaveSendBootUp()

```
void slaveSendBootUp ( )
```

This function sends a slave boot-up message over CAN servizi network.

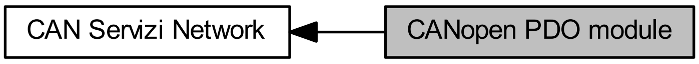
Author

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

Definition at line 59 of file nmt.cpp.

7.10 CANopen PDO module

Collaboration diagram for CANopen PDO module:



Functions

- void buildPDO (uint8_t PDOtype, Message ∗pdo)

  This function serializes data to send into PDO message.
- void proceedPDO (Message ∗pdo)

  This function manages PDO message receive, deserializing data.

7.10.1 Detailed Description

Data into PDOs are configured according TPDO_configuration.

7.10.2 Function Documentation

7.10.2.1 buildPDO()

```
void buildPDO (
            uint8_t PDOtype,
            Message * pdo )
```

This function serializes data to send into PDO message.

Author

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

Parameters

in	*PDOtype*	PDO type according to CANopen Function Codes.
in	*pdo*	CANopen message to build

Definition at line 19 of file pdo.cpp.

7.10.2.2 proceedPDO()

```
void proceedPDO (
            Message * pdo )
```

This function manages PDO message receive, deserializing data.

Author

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

Parameters

in	*pdo*	CANopen message to manage

Definition at line 56 of file pdo.cpp.

7.11 Main module

Functions

- void setup ()

  This function perform basic board setup. Upon power-up SCU (CANopen slave node) goes into initialization. It initializes the entire application, CAN/CANopen interfaces and communication. At the end of the initialization the node tries to transmit boot-up message. As soon as it is transmitted successfully, the node switches to Pre-operational state.
- void loop ()

  This function is called into endless while main loop. It takes care of sending data through radio, if enabled.

7.11.1 Detailed Description

7.11.2 Function Documentation

7.11.2.1 loop()

```
void loop ( )
```

This function is called into endless while main loop. It takes care of sending data through radio, if enabled.

Author

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

Definition at line 90 of file SCU.ino.

7.11.2.2 setup()

```
void setup ( )
```

This function perform basic board setup. Upon power-up SCU (CANopen slave node) goes into initialization. It initializes the entire application, CAN/CANopen interfaces and communication. At the end of the initialization the node tries to transmit boot-up message. As soon as it is transmitted successfully, the node switches to Pre-operational state.

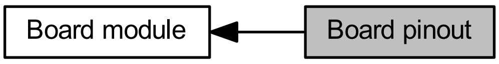
Author

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

Definition at line 70 of file SCU.ino.

7.12 Board pinout

Collaboration diagram for Board pinout:



Macros

- #define CAN_PORT Can0

  Pin on board dedicated to CAN port.
- #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 FR_SX_SUSP_PIN A3

  Pin on board dedicated to frontal left suspension sensor.
- #define FR_SX_ADC_CHAN_NUM ADC_CHER_CH4

  GPIO pin on the Atmel SAM3X8E processor corresponding to frontal left suspension signal (AD4)
- #define FR_DX_SUSP_PIN A4

  Pin on board dedicated to frontal right suspension sensor.
- #define FR_DX_ADC_CHAN_NUM ADC_CHER_CH3

  GPIO pin on the Atmel SAM3X8E processor corresponding to frontal right suspension signal (AD3)
- #define FR_SX_PW_PIN 36

  Pin on board dedicated to frontal left phonic wheel encoder.
- #define FR_DX_PW_PIN 38

  Pin on board dedicated to frontal right phonic wheel encoder.

7.12.1 Detailed Description

Board pinout for each sensor, CAN port and radio.

Chapter 8

Class Documentation

8.1 Message Struct Reference

CANopen message struct.

```
#include <CO_can.h>
```

Public Attributes

- uint16_t cob_id
- uint8_t len
- uint8_t data [8]

8.1.1 Detailed Description

CANopen message struct.

Definition at line 90 of file CO_can.h.

8.1.2 Member Data Documentation

8.1.2.1 cob_id

```
uint16_t Message::cob_id
```

message's COB-ID

Definition at line 91 of file CO_can.h.

8.1.2.2 data

```
uint8_t Message::data[8]
```

message's datas

Definition at line 93 of file CO_can.h.

8.1.2.3 len

```
uint8_t Message::len
```

message's length (0 to 8)

Definition at line 92 of file CO_can.h.

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

- CO_can.h

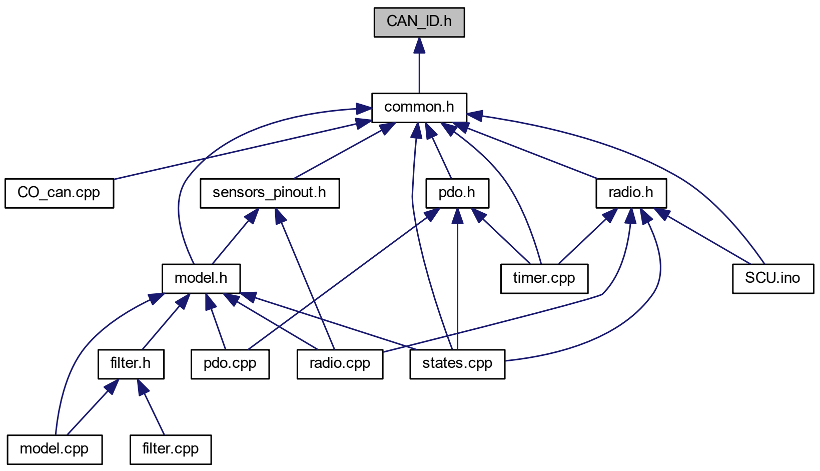
Chapter 9

File Documentation

9.1 CAN_ID.h File Reference

CAN servizi nodeIDs module header.

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



Macros

- #define SCU_FRONTAL_NODE_ID 1

  Frontal SCU node ID on CAN servizi network.
- #define VCU_NODE_ID 2

  VCU node ID on CAN servizi network.
- #define SCU_REAR_NODE_ID 3

  Rear SCU node ID on CAN servizi network.
- #define TCU_NODE_ID 4

  TCU node ID on CAN servizi network.

9.1.1 Detailed Description

CAN servizi nodeIDs module header.

Author

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

Date

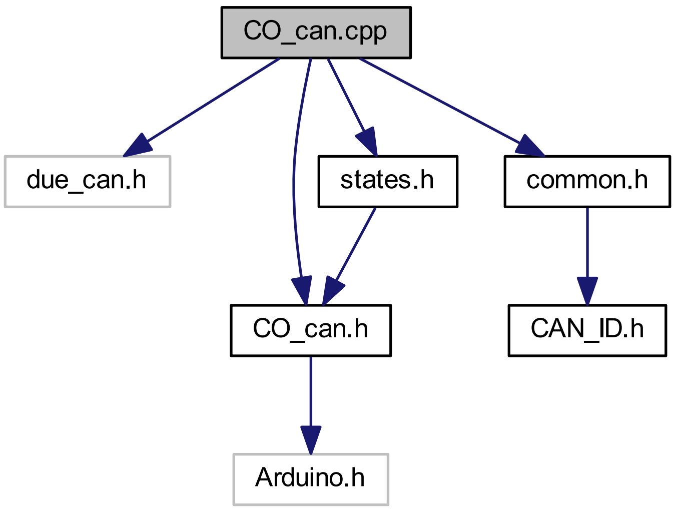
2018

9.2 CO_can.cpp File Reference

CANOpen main module implementation file.

```
#include <due_can.h>
#include "CO_can.h"
#include "states.h"
#include "common.h"
```

Include dependency graph for CO_can.cpp:



Functions

- void canSend (Message *m)

  This function send a CANopen message over CAN servizi network.
- void **CAN_general_callback** (CAN_FRAME *frame)
- void initCAN ()

  This function initialize CAN/CANopen interfaces and communication.

9.2.1 Detailed Description

CANOpen main module implementation file.

Author

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

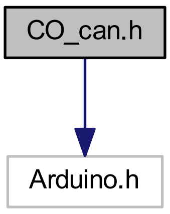
Date

2018

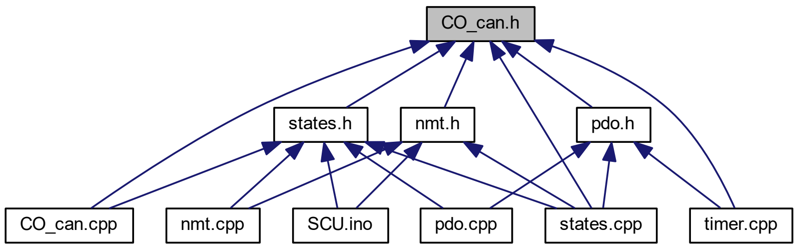
9.3 CO_can.h File Reference

CANOpen main module header.

```
#include <Arduino.h>
```
Include dependency graph for CO_can.h:



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



Classes

- struct Message

  CANopen message struct.

Macros

- #define Message_Initializer {0,0,{0,0,0,0,0,0,0,0}}

  CANopen static message initializer.

Functions

- void initCAN ()

  This function initialize CAN/CANopen interfaces and communication.
- void canSend (Message ∗m)

  This function send a CANopen message over CAN servizi network.

9.3.1 Detailed Description

CANOpen main module header.

Author

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

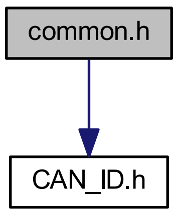
Date

2018

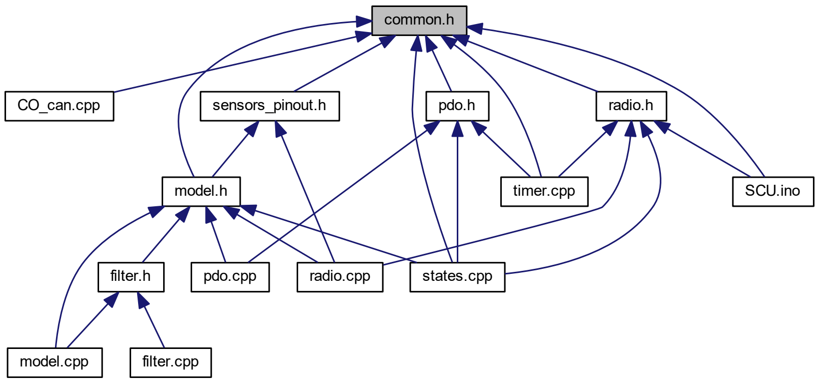
9.4 common.h File Reference

This file contains some common macro definitions for configuring main relevants parameters.

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



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



Macros

- #define CAN_BAUDRATE 1000000

  CAN network baud rate [bps].
- #define SERIAL_BAUDRATE 115200

  Serial UART baud rate [bps].
- #define TIME_SLOT_PERIOD 4000

  Timer period [ms].
- #define TIMER Timer3

  CANopen timer.
- #define _FRONTAL_

  Macro for selecting frontal SCU firmware.
- #define _RETRO_

  Macro for selecting rear SCU firmware.

9.4.1 Detailed Description

This file contains some common macro definitions for configuring main relevants parameters.

Author

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

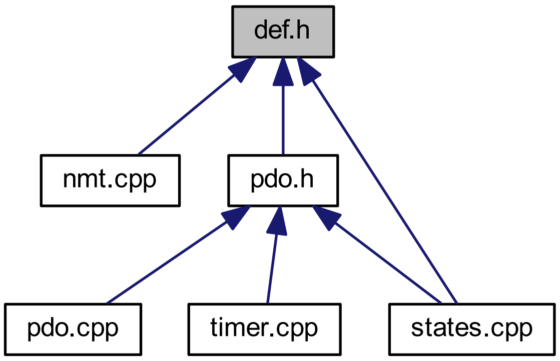
Date

2018

9.5 def.h File Reference

CANopen DS301 definitions.

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



Macros

- #define NMT 0x0

  NMT function code.
- #define SYNC 0x1

  SYNC function code.
- #define TIME_STAMP 0x2

  TIME_STAMP function code.
- #define PDO1tx 0x3

  PDO1tx function code.
- #define PDO1rx 0x4

  PDO1rx function code.
- #define PDO2tx 0x5

  PDO2tx function code.
- #define PDO2rx 0x6

  PDO2rx function code.
- #define PDO3tx 0x7

  PDO3tx function code.
- #define PDO3rx 0x8

  PDO3rx function code.
- #define PDO4tx 0x9

  PDO4tx function code.
- #define PDO4rx 0xA

  PDO4rx function code.
- #define SDOtx 0xB

  SDOtx function code.
- #define SDOrx 0xC

SDOrx function code.

- #define NODE_GUARD 0xE

  NODE GUARD function code.

- #define LSS 0xF

  LSS function code.

- #define GET_FUNC_CODE(COB_ID) (COB_ID >> 7)

  Extract function code from COB-ID.

- #define SET_FUNC_CODE(COB_ID) (COB_ID << 7)

  Set function code to COB-ID.

- #define NMT_Start_Node 0x01

  'go Operational' command specifier

- #define NMT_Stop_Node 0x02

  'stop Node' command specifier

- #define NMT_Enter_PreOperational 0x80

  'go PreOperational' command specifier

- #define NMT_Reset_Node 0x81

  'reset Node' command specifier

- #define NMT_Reset_Comunication 0x82

  'reset Communication' command specifier

9.5.1 Detailed Description

CANopen DS301 definitions.

Author

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

Date

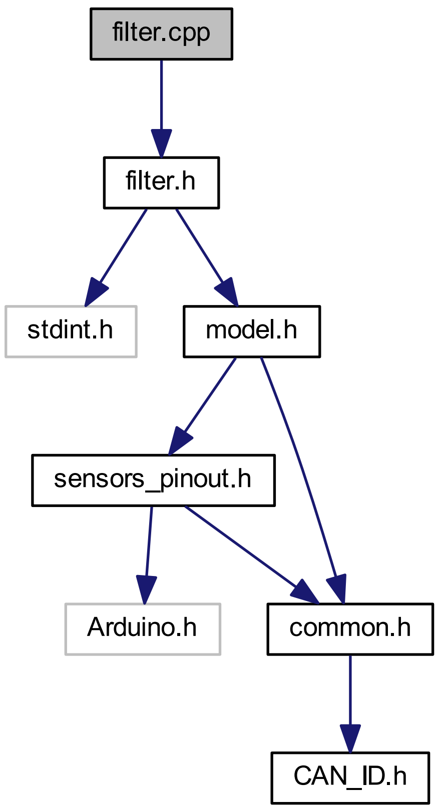
2018

9.6 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.

9.6.1 Detailed Description

Filter module implementation file.

Author

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

Date

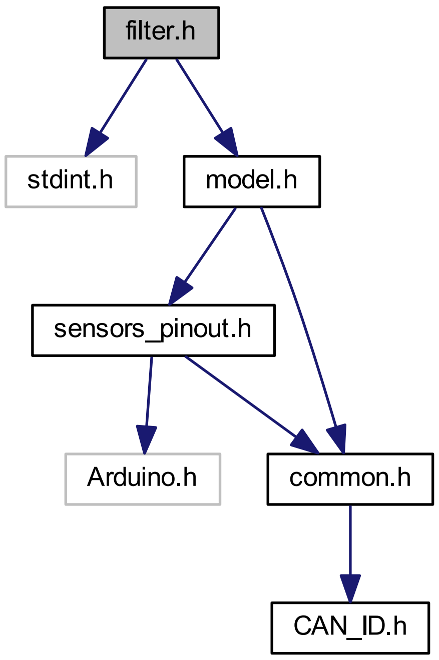
2018

9.7 filter.h File Reference

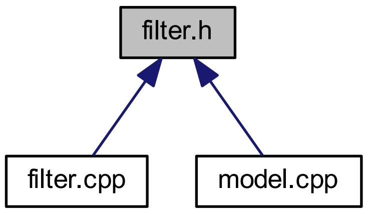
Filter module header file.

```
#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.

9.7.1 Detailed Description

Filter module header file.

Author

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

Date

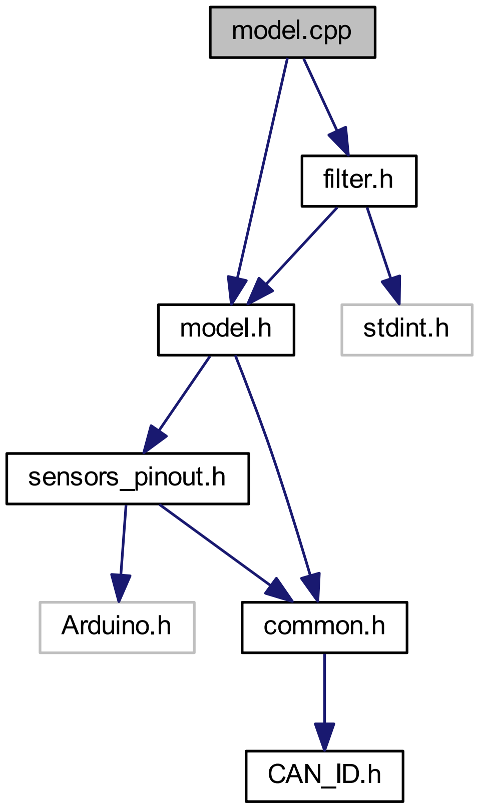
2018

9.8 model.cpp File Reference

Board model implementation file.

```
#include "model.h"
#include "filter.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 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)
- #define SUSPENSIONS_MIN 0

  Minimum suspension stroke [mm].
- #define SUSPENSIONS_ADC_MAX ADC_MAX

  Maximum suspension sensor V_{OUT}.
- #define SUSP_STROKE_NORMALIZE (SUSP_STROKE_EXTENSION / SUSPENSIONS_ADC_MAX)

  Suspension stroke voltage normalizer.
- #define COGS_NUMBER 30.0d

  Number of phonic wheel's cogs.
- #define NORMALIZE_RPM 1000000.0d

  Normalize time domain [μs].
- #define RPM_MIN 10

  Rpm lower bound under that rpm are forced to zero.
- #define ACCELEROMETER_MAX_G 5.0d

  Accelerometer sensor maximum value [m/s^2].
- #define ACCELEROMETER_NORMALIZE 2.0d $*$ ACCELEROMETER_MAX_G / ADC_MAX

  Accelerometer sensor voltage normalizer.
- #define APPS_PLAUS_RANGE 10

  Maximum percentage deviation of pedal travel between two APPS.
- #define SCU_FRONTAL_ADC_CHANNELS 5

  Number of ADC channels used in frontal SCU board.
- #define SCU_FRONTAL_ADC_CHANNELS_LIST TPS1_ADC_CHAN_NUM | TPS2_ADC_CHAN_NUM | BRAKE_ADC_CHAN_NUM | FR_SX_ADC_CHAN_NUM | FR_DX_ADC_CHAN_NUM

  List of ADC channels dedicated to each IO port in frontal SCU board.
- #define SCU_RETRO_ADC_CHANNELS 4

  Number of ADC channels used in rear SCU board.
- #define SCU_RETRO_ADC_CHANNELS_LIST ACC_X_ADC_CHAN_NUM | ACC_Z_ADC_CHAN_NUM | RT_SX_ADC_CHAN_NUM | RT_DX_ADC_CHAN_NUM

  List of ADC channels dedicated to each IO port in retro SCU board.
- #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 FR_SX_ADC_OFFSET 3

  Offset from DMA buffer.
- #define FR_DX_ADC_OFFSET 4

  Offset from DMA buffer.
- #define ACC_X_ADC_OFFSET 0

  Offset from DMA buffer.
- #define ACC_Z_ADC_OFFSET 1

  Offset from DMA buffer.
- #define RT_SX_ADC_OFFSET 2

  Offset from DMA buffer.
- #define RT_DX_ADC_OFFSET 3

  Offset from DMA buffer.
- #define BUFFER_LENGTH ADC_BUFFER_SIZE $*$ ADC_CHANNELS

  Length, in bytes, of each DMA buffer.

Functions

- void fr_sx_pulse ()

  EXTI IRQ handler. External interrupt handler executed when frontal left wheel encoder finds a hole into phonic wheel.
- void fr_dx_pulse ()

  EXTI IRQ handler. External interrupt handler executed when frontal right wheel encoder finds a hole into phonic wheel.
- volatile uint16_t get_fr_sx_rpm ()

  If rpm value is lower than RPM_MIN, output is forced to zero.
- volatile uint16_t get_fr_dx_rpm ()

  If rpm value is lower than RPM_MIN, output is forced to zero.
- 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.

Variables

- volatile int bufn

  DMA buffer pointer.
- volatile int obufn

  DMA buffer pointer.
- volatile uint16_t buf [BUFFERS][BUFFER_LENGTH]

  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.
- 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 tps1_max = TPS1_UPPER_BOUND

  First APPS max output voltage (equals to TPS1_UPPER_BOUND)
- volatile uint16_t tps1_min = TPS1_LOWER_BOUND

  First APPS min output voltage (equals to TPS1_LOWER_BOUND)
- volatile uint16_t tps2_max = TPS2_UPPER_BOUND

  Second APPS max output voltage (equals to TPS2_UPPER_BOUND)
- volatile uint16_t tps2_min = TPS2_LOWER_BOUND

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

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

  Brake sensor min output voltage (equals to BRAKE_LOWER_BOUND)
- volatile uint8_t tps1_percentage = 0

  First APPS percentage value retrieved by tps1 signal (TPS1_PIN)
- volatile uint8_t tps2_percentage = 0

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

  Brake pedal position sensor percentage value retrieved by brake signal (BRAKE_PIN)

- volatile bool apps_plausibility = true

  APPS plausibility status.
- volatile bool brake_plausibility = true

  Brake plausibility status.
- volatile uint8_t fr_sx_susp

  Frontal left suspension stroke [mm].
- volatile uint8_t fr_dx_susp

  Frontal right suspension stroke [mm].
- volatile uint16_t fr_sx_rpm = 0

  Frontal left wheel velocity [rpm].
- volatile uint16_t fr_dx_rpm = 0

  Frontal right wheel velocity [rpm].
- volatile unsigned long fr_sx_prev

  Frontal left wheel encoder previous timestamp.
- volatile unsigned long fr_sx_curr

  Frontal left wheel encoder current timestamp.
- volatile unsigned long fr_dx_prev

  Frontal right wheel encoder previous timestamp.
- volatile unsigned long fr_dx_curr

  Frontal right wheel encoder current timestamp.

9.8.1 Detailed Description

Board model implementation file.

Author

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

Date

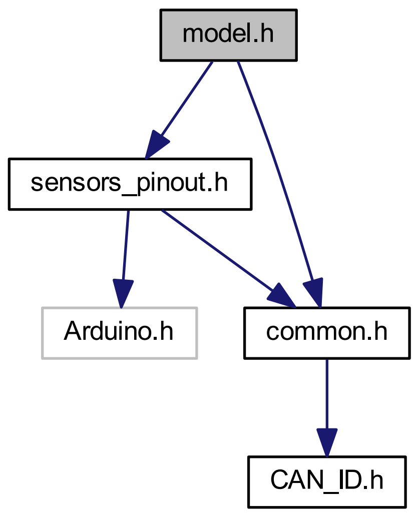
2018

9.9 model.h File Reference

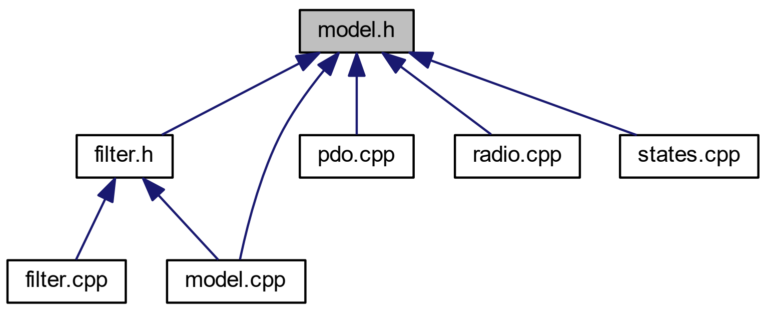
Board model header file.

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

Include dependency graph for model.h:



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



Macros

- #define SUSP_STROKE_EXTENSION 75.0

    Maximum suspension stroke [mm].

Functions

- volatile uint16_t get_fr_sx_rpm ()

    If rpm value is lower than RPM_MIN, output is forced to zero.

- volatile uint16_t get_fr_dx_rpm ()

  If rpm value is lower than RPM_MIN, output is forced to zero.
- volatile uint16_t get_rt_sx_rpm ()

  This function returns retro left wheel velocity [rpm].
- volatile uint16_t get_rt_dx_rpm ()

  This function returns retro right wheel velocity [rpm].
- void model_init ()

  This function initializes hardware board.

Variables

- volatile uint8_t tps1_percentage

  First APPS percentage value retrieved by tps1 signal (TPS1_PIN)
- volatile uint8_t tps2_percentage

  Second APPS percentage value retrieved by tps2 signal (TPS2_PIN)
- volatile uint8_t brake_percentage

  Brake pedal position sensor percentage value retrieved by brake signal (BRAKE_PIN)
- volatile bool apps_plausibility

  APPS plausibility status.
- volatile bool brake_plausibility

  Brake plausibility status.
- volatile uint8_t fr_sx_susp

  Frontal left suspension stroke [mm].
- volatile uint8_t fr_dx_susp

  Frontal right suspension stroke [mm].
- volatile uint8_t acc_x_value

  Accelerometer X value [m/s^2].
- volatile uint8_t acc_z_value

  Accelerometer Z value [m/s^2].
- volatile uint8_t rt_sx_susp

  Retro left suspension stroke [mm].
- volatile uint8_t rt_dx_susp

  Retro right suspension stroke [mm].

9.9.1 Detailed Description

Board model header file.

Author

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

Date

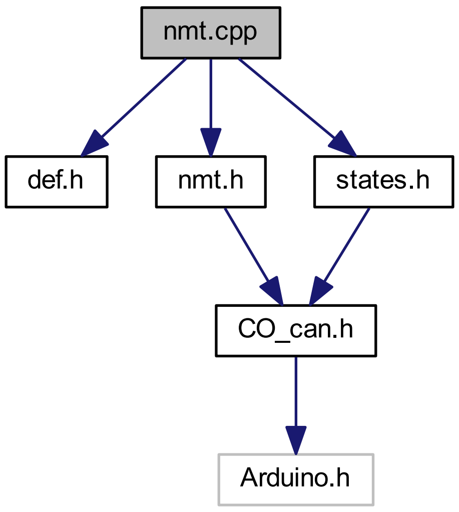
2018

9.10 nmt.cpp File Reference

CANOpen NMT module implementation file.

```
#include "def.h"
#include "nmt.h"
#include "states.h"
```

Include dependency graph for nmt.cpp:



Functions

- void proceedNMTstateChange (Message ∗m)

  According to CANopen NMT Command Specifiers, upon NMT reception from VCU master node, SCU change current state.
- void slaveSendBootUp ()

  This function sends a slave boot-up message over CAN servizi network.

9.10.1 Detailed Description

CANOpen NMT module implementation file.

Author

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

Date

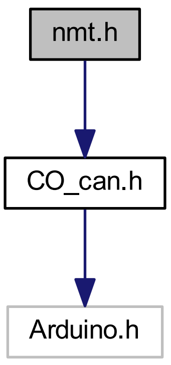
2018

9.11 nmt.h File Reference

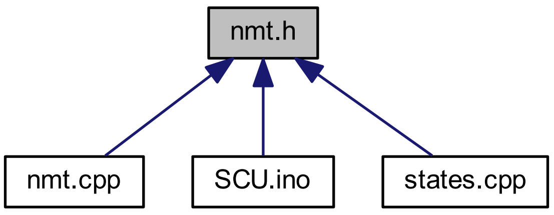
CANOpen NMT module header.

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

Include dependency graph for nmt.h:



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



Functions

- void proceedNMTstateChange (Message ∗m)

  According to CANopen NMT Command Specifiers, upon NMT reception from VCU master node, SCU change current state.

- void slaveSendBootUp ()

  This function sends a slave boot-up message over CAN servizi network.

9.11.1 Detailed Description

CANOpen NMT module header.

Author

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

Date

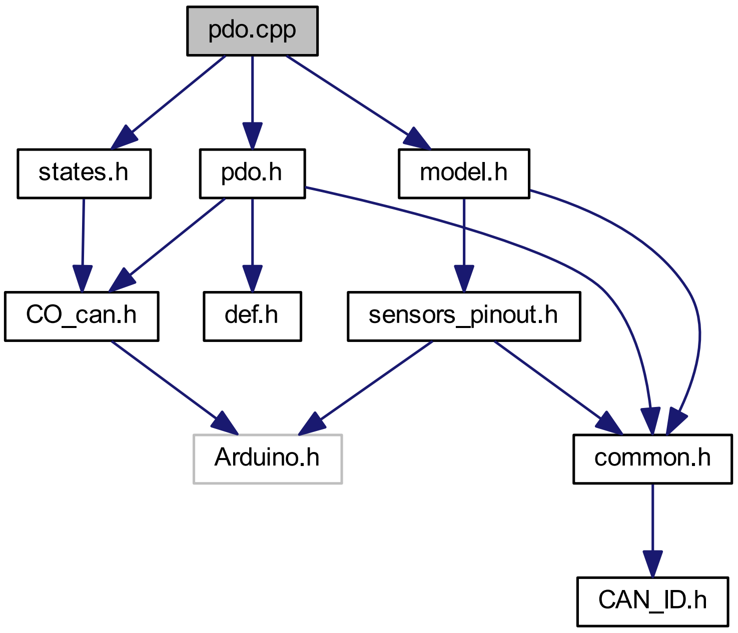
2018

9.12 pdo.cpp File Reference

CANopen PDO support header file.

```
#include "pdo.h"
#include "states.h"
#include "model.h"
```

Include dependency graph for pdo.cpp:



Functions

- void buildPDO (uint8_t PDOtype, Message ∗pdo)

  This function serializes data to send into PDO message.
- void proceedPDO (Message ∗pdo)

  This function manages PDO message receive, deserializing data.

9.12.1 Detailed Description

CANopen PDO support header file.

Author

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

Date

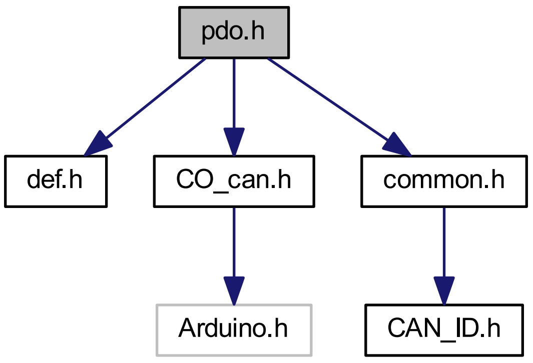
2018

9.13 pdo.h File Reference

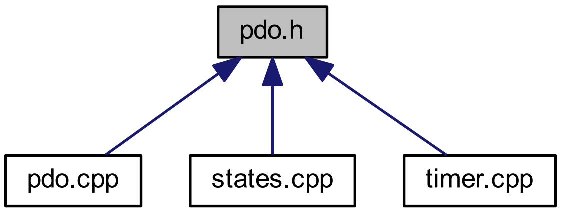
CANopen PDO support header file.

```
#include "def.h"
#include "CO_can.h"
#include "common.h"
```

Include dependency graph for pdo.h:



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



Functions

- void buildPDO (uint8_t PDOtype, Message ∗pdo)

  This function serializes data to send into PDO message.
- void proceedPDO (Message ∗pdo)

  This function manages PDO message receive, deserializing data.

9.13.1 Detailed Description

CANopen PDO support header file.

Author

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

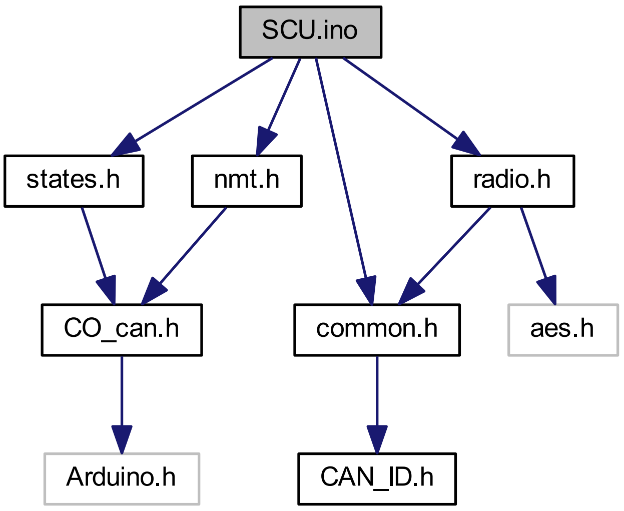
Date

2018

9.14 SCU.ino File Reference

Main module file.

```
#include "common.h"
#include "states.h"
#include "nmt.h"
#include "radio.h"
```
Include dependency graph for SCU.ino:



Functions

- void setup ()

  This function perform basic board setup. Upon power-up SCU (CANopen slave node) goes into initialization. It initializes the entire application, CAN/CANopen interfaces and communication. At the end of the initialization the node tries to transmit boot-up message. As soon as it is transmitted successfully, the node switches to Pre-operational state.
- void loop ()

  This function is called into endless while main loop. It takes care of sending data through radio, if enabled.

9.14.1 Detailed Description

Main module file.

Author

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

Date

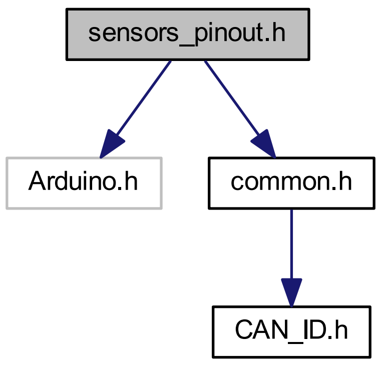
2018

9.15 sensors_pinout.h File Reference

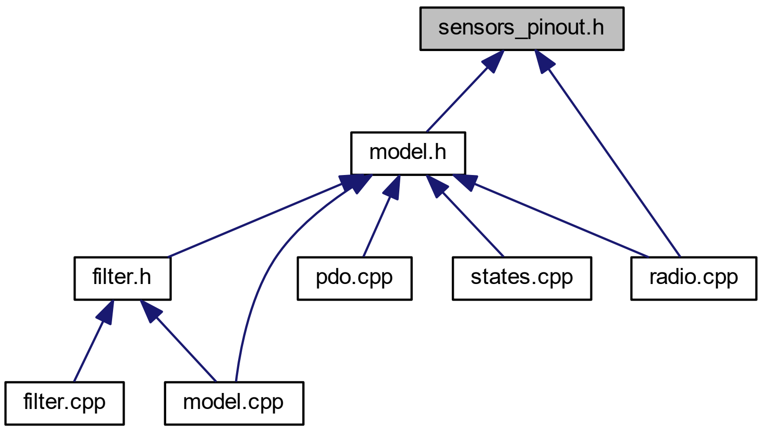
Board pinout module header.

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

Include dependency graph for sensors_pinout.h:



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



Macros

- #define CAN_PORT Can0

  Pin on board dedicated to CAN port.
- #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 FR_SX_SUSP_PIN A3

  Pin on board dedicated to frontal left suspension sensor.
- #define FR_SX_ADC_CHAN_NUM ADC_CHER_CH4

  GPIO pin on the Atmel SAM3X8E processor corresponding to frontal left suspension signal (AD4)
- #define FR_DX_SUSP_PIN A4

  Pin on board dedicated to frontal right suspension sensor.
- #define FR_DX_ADC_CHAN_NUM ADC_CHER_CH3

  GPIO pin on the Atmel SAM3X8E processor corresponding to frontal right suspension signal (AD3)
- #define FR_SX_PW_PIN 36

  Pin on board dedicated to frontal left phonic wheel encoder.
- #define FR_DX_PW_PIN 38

  Pin on board dedicated to frontal right phonic wheel encoder.

9.15.1 Detailed Description

Board pinout module header.

Author

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

Date

2018

Index