Sensors Control Unit (SCU)

Firmware Specifications

History

Version	Author	Date
1.0	Arella Matteo	2018

Contents

1	Fast	Charge	SAE SCU	I firmware	1
2	CAN	l Serviz	i network		3
3	Boa	rd mode	el		5
4	Rea	I Time T	elemetry	System	7
5	Finit	te State	Machine	(FSM)	9
6	Mod	lule Inde	ex		11
	6.1	Module	es		. 11
7	Clas	s Index			13
	7.1	Class	List		. 13
8	File	Index			15
	8.1	File Lis	st		. 15
9	Mod	lule Doc	umentati	on	17
	9.1	CAN S	ervizi Net	work	. 17
		9.1.1	Detailed	Description	. 18
		9.1.2	Function	Documentation	. 18
			9.1.2.1	CAN_general_callback()	. 18
			9.1.2.2	canSend()	. 19
			9.1.2.3	getNodeld()	. 19
			9.1.2.4	initCAN()	. 20
			9.1.2.5	setNodeld()	. 20

iv CONTENTS

9.2	CAN N	etwork Nodes ID		 	 	 	21
	9.2.1	Detailed Descrip	otion	 	 	 	21
9.3	Comm	on Defines		 	 	 	22
	9.3.1	Detailed Descrip	otion	 	 	 	22
9.4	SCU fi	rmware selection		 	 	 	23
	9.4.1	Detailed Descrip	otion	 	 	 	23
9.5	CANop	en Function Cod	es	 	 	 	24
	9.5.1	Detailed Descrip	otion	 	 	 	24
9.6	CANop	en NMT Comma	nd Specifiers	 	 	 	25
	9.6.1	Detailed Descrip	otion	 	 	 	25
9.7	Data fi	tering module .		 	 	 	26
	9.7.1	Detailed Descrip	otion	 	 	 	26
	9.7.2	Function Docum	nentation	 	 	 	26
		9.7.2.1 filter_	buffer()	 	 	 	26
9.8	Board	module		 	 	 	28
	9.8.1	Detailed Descrip	otion	 	 	 	32
	9.8.2	Macro Definition	Documentation	 	 	 	32
		9.8.2.1 BUFF	ERS	 	 	 	32
	9.8.3	Function Docum	nentation	 	 	 	32
		9.8.3.1 ADC_	Handler()	 	 	 	32
		9.8.3.2 fr_dx_	_pulse()	 	 	 	32
		9.8.3.3 fr_sx_	_pulse()	 	 	 	33
		9.8.3.4 get_fr	dx_rpm()	 	 	 	33
		9.8.3.5 get_fr	_sx_rpm()	 	 	 	33
		9.8.3.6 get_rt	:_dx_rpm()	 	 	 	34
		9.8.3.7 get_rt	:_sx_rpm()	 	 	 	34
		9.8.3.8 mode	I_init()	 	 	 	34
	9.8.4	Variable Docum	entation	 	 	 	35
		9.8.4.1 apps_	plausibility [1/2]	 	 	 	35
		9.8.4.2 apps_	plausibility [2/2]	 	 	 	35

CONTENTS

		9.8.4.3	brake_percentage [1/2]	35
		9.8.4.4	brake_percentage [2/2]	35
		9.8.4.5	brake_plausibility [1/2]	36
		9.8.4.6	brake_plausibility [2/2]	36
		9.8.4.7	tps1_percentage [1/2]	36
		9.8.4.8	tps1_percentage [2/2]	36
		9.8.4.9	tps2_percentage [1/2]	36
		9.8.4.10	tps2_percentage [2/2]	36
9.9	CANop	en NMT n	nodule	37
	9.9.1	Detailed	Description	37
	9.9.2	Function	Documentation	37
		9.9.2.1	proceedNMTstateChange()	37
		9.9.2.2	slaveSendBootUp()	38
9.10	CANop	en PDO m	nodule	39
	9.10.1	Detailed	Description	39
	9.10.2	Function	Documentation	39
		9.10.2.1	buildPDO()	39
		9.10.2.2	proceedPDO()	40
9.11	Radio	module .		41
	9.11.1	Detailed	Description	42
	9.11.2	Macro De	efinition Documentation	42
		9.11.2.1	CIPHER_MAX_LENGTH	42
		9.11.2.2	JSON_BUFFER_SIZE	42
	9.11.3	Function	Documentation	42
		9.11.3.1	byte_padding()	42
		9.11.3.2	encrypt_model()	43
		9.11.3.3	generate_iv()	43
		9.11.3.4	generate_random_char()	44
		9.11.3.5	pkcs7_padding()	44
		9.11.3.6	radio_init()	45

vi

		9.11.3.7 radio_send_model()	45
	9.11.4	Variable Documentation	45
		9.11.4.1 iv	45
9.12	Main m	odule	46
	9.12.1	Detailed Description	46
	9.12.2	Function Documentation	46
		9.12.2.1 loop()	46
		9.12.2.2 setup()	46
9.13	Board p	pinout	47
	9.13.1	Detailed Description	47
9.14	CANop	en Finite State Machine module	48
	9.14.1	Detailed Description	49
	9.14.2	Typedef Documentation	49
		9.14.2.1 e_nodeState	49
	9.14.3	Enumeration Type Documentation	49
		9.14.3.1 enum_nodeState	49
	9.14.4	Function Documentation	49
		9.14.4.1 canDispatch()	49
		9.14.4.2 getState()	50
		9.14.4.3 initialisation()	50
		9.14.4.4 operational()	51
		9.14.4.5 preOperational()	51
		9.14.4.6 setState()	51
		9.14.4.7 stopped()	52
9.15	CANop	en Timer module	53
	9.15.1	Detailed Description	54
	9.15.2	Function Documentation	54
		9.15.2.1 TimeDispatch()	54
		9.15.2.2 timerInit()	54
		9.15.2.3 timerStart()	54
		9.15.2.4 timerStop()	54

CONTENTS vii

10	Clas	s Documentation	55
	10.1	Message Struct Reference	55
		10.1.1 Detailed Description	55
		10.1.2 Member Data Documentation	55
		10.1.2.1 cob_id	55
		10.1.2.2 data	56
		10.1.2.3 len	56
11	File I	Documentation Company of the Company	57
	11.1	CAN_ID.h File Reference	57
		11.1.1 Detailed Description	58
	11.2	CO_can.cpp File Reference	58
		11.2.1 Detailed Description	59
	11.3	CO_can.h File Reference	60
		11.3.1 Detailed Description	61
	11.4	common.h File Reference	61
		11.4.1 Detailed Description	62
	11.5	def.h File Reference	63
		11.5.1 Detailed Description	64
	11.6	filter.cpp File Reference	64
		11.6.1 Detailed Description	65
	11.7	filter.h File Reference	66
		11.7.1 Detailed Description	67
	11.8	model.cpp File Reference	67
		11.8.1 Detailed Description	71
	11.9	model.h File Reference	71
		11.9.1 Detailed Description	73
	11.10	Onmt.cpp File Reference	74
		11.10.1 Detailed Description	74
	11.11	nmt.h File Reference	75
		11.11.1 Detailed Description	76

viii CONTENTS

	11.12pdo.cpp File Reference	76
	11.12.1 Detailed Description	77
	11.13pdo.h File Reference	77
	11.13.1 Detailed Description	78
	11.14radio.cpp File Reference	78
	11.14.1 Detailed Description	80
	11.15radio.h File Reference	80
	11.15.1 Detailed Description	81
	11.16SCU.ino File Reference	82
	11.16.1 Detailed Description	82
	11.17sensors_pinout.h File Reference	83
	11.17.1 Detailed Description	84
	11.18states.cpp File Reference	84
	11.18.1 Detailed Description	85
	11.19states.h File Reference	86
	11.19.1 Detailed Description	87
	11.20timer.cpp File Reference	88
	11.20.1 Detailed Description	89
	11.21timer.h File Reference	89
	11.21.1 Detailed Description	90
Inc	dex	91
HIL	AGA	<i>3</i> I

FastChargeSAE SCU firmware

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

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.

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

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

4 CAN Servizi network

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 Off- set	#CAN packet	ID	Total Length
	First APPS	%	$1\ Byte$	[0:7]			
	Second A← PPS	%	1~Byte	[8:15]	#1	TPDO1 +	4
	Brake	%	1 Byte	[16:23]		NODE-ID	
SCU_{FRONT}	APPS plau- sibility	bool	4 bit	[24:27]			
	Brake plau- sibility	bool	4 bit	[28:31]			
	Right phonic wheel	rpm	2 Bytes	[0:15]			
	Left phonic wheel	rpm	2 Bytes	[16:31]			
	Right sus- pension	mm	1~Byte	[32:39]	#2	TPDO2 + NODE-ID	6
	Left sus- pension	mm	1~Byte	[40:47]			
	Accel. X	m/s^2	1 Byte	[0:7]			
	Accel. Z	m/s^2	1 Byte	[8:15]			
SCU_{REAR}	Right sus- pension	mm	1 Byte	[16:23]	#3	TPDO1 + NODE-ID	8
	Left sus- pension	mm	1 Byte	[24:31]			
	Right phonic wheel	rpm	2 Bytes	[32:47]			
	Left phonic wheel	rpm	$2\ Bytes$	[48:63]			
TCU	Torque lim- iter	%	1 Byte	[0:7]	#4	TPDO1 + NODE-ID	1

where TPDO1 = 0x180 and TPDO2 = 0x280

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

Board pinout is described in Board module.

Output rpm are returned following this formula:

$$rpm = NORMALIZE_RPM * (\frac{60}{COGS_NUMBER * \Delta_{TIMESTAMP}})$$

where $NORMALIZE_RPM$ normalizes TIMESTAMP in seconds.

6 Board model

Real Time Telemetry System

A real time telemetry system was implemented by a radio bridge. SCU_{REAR} is connected to a radio transmission module, which sends the telemetry data to a receiving radio module placed at the boxes. The data is serialized in JSON format and encrypted using AES with a 192bit key (in CTR mode). The ciphertext thus obtained is coded in base64 and sent via radio. In reception, the text is decoded and decrypted and sent via serial port to a computer. The data thus obtained are deserialized by a multiplatform application developed in JavaFX, which has the ability to graph the values of the various sensors in real time and to save them in log files to be analyzed later.

Finite State Machine (FSM)

Each SCU board can be represented by a finite state machine with the following statuses: Initialisation, Preoperational, 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 preoperational 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.

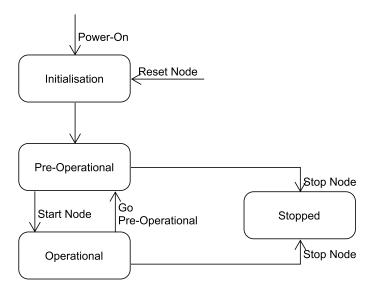


Figure 5.1 FSM diagram

Module Index

6.1 Modules

Here is a list of all modules:

CAN Servizi Network
CAN Network Nodes ID
CANopen Function Codes
CANopen NMT module
CANopen NMT Command Specifiers
CANopen PDO module
CANopen Finite State Machine module
CANopen Timer module
Common Defines
SCU firmware selection
Board module
Data filtering module
Board pinout
Radio module
Main module

12 Module Index

Class Index

7	4	Class	List
/		U.IAC	z i iet

Here are the classes, structs, unions and interfaces with brief description	s:
Message	

14 Class Index

File Index

8.1 File List

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

CAN_ID.h	
CAN servizi nodeIDs module header	57
CO_can.cpp	
CANOpen main module implementation file	58
CO_can.h	
CANOpen main module header	60
common.h	
This file contains some common macro definitions for configuring main relevants parameters .	61
def.h	
CANopen DS301 definitions	63
filter.cpp	0.4
Filter module implementation file	64
filter.h	00
Filter module header file	66
model.cpp	67
Board model implementation file	07
Board model header file	71
nmt.cpp	/ 1
CANOpen NMT module implementation file	74
nmt.h	/ -
CANOpen NMT module header	75
pdo.cpp	. •
CANopen PDO support header file	76
pdo.h	
CANopen PDO support header file	77
radio.cpp	
Radio implementation file	78
radio.h	
Radio header file	80
SCU.ino	
Main module file	82
sensors_pinout.h	
Board pinout module header	83
states.cpp	
CANopen finite state machine implementation file	84

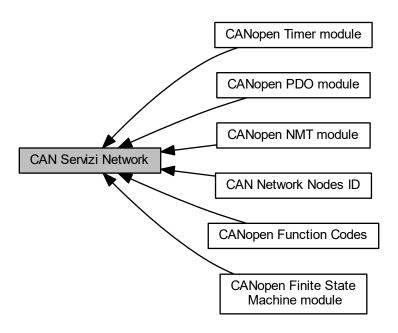
16 File Index

states.h		
	CANopen finite state machine header file	86
timer.cpp		
	CANopen timer implementation file	88
timer.h		
	CANopen timer header file	89

Module Documentation

9.1 CAN Servizi Network

Collaboration diagram for CAN Servizi Network:



Modules

- CAN Network Nodes ID
- CANopen Function Codes
- CANopen NMT module
- CANopen PDO module
- CANopen Finite State Machine module
- CANopen Timer module

18 Module Documentation

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

• uint8_t getNodeld ()

This function returns node ID into CAN servizi network.

void setNodeld (uint8 t nodeld)

This function sets node ID into CAN servizi network.

void canSend (Message *m)

This function send a CANopen message over CAN servizi network.

• void CAN_general_callback (CAN_FRAME *frame)

This function manage CAN frame receiption and dispatch CANopen message.

• void initCAN ()

This function initialize CAN/CANopen interfaces and communication.

Variables

· uint8 t nodeld

Board node ID into CAN servizi network according to NODE-ID.

- 9.1.1 Detailed Description
- 9.1.2 Function Documentation

```
9.1.2.1 CAN_general_callback()
```

This function manage CAN frame receiption and dispatch CANopen message.

Author

```
Arella Matteo
```

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

9.1 CAN Servizi Network

Parameters

in	frame	CAN received frame.

Definition at line 57 of file CO_can.cpp.

9.1.2.2 canSend()

This function send a CANopen message over CAN servizi network.

Author

Arella Matteo

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

Parameters

	in n	n	CANopen message to send
--	--------	---	-------------------------

Definition at line 37 of file CO_can.cpp.

9.1.2.3 getNodeld()

```
uint8_t getNodeId ( )
```

This function returns node ID into CAN servizi network.

Author

Arella Mattec

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

Returns

CANopen node ID

Definition at line 28 of file CO_can.cpp.

20 Module Documentation

9.1.2.4 initCAN()

```
void initCAN ( )
```

This function initialize CAN/CANopen interfaces and communication.

Author

```
Arella Matteo
```

```
(mail: arella.1646983@studenti.uniromal.it)
```

Definition at line 68 of file CO_can.cpp.

9.1.2.5 setNodeld()

This function sets node ID into CAN servizi network.

Author

Arella Matteo

```
(mail: arella.1646983@studenti.uniromal.it)
```

Parameters

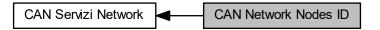
in	node←	CANopen node ID
	ld	

Definition at line 33 of file CO_can.cpp.

9.2 CAN Network Nodes ID 21

9.2 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

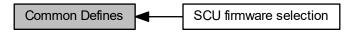
TCU node ID on CAN servizi network.

9.2.1 Detailed Description

22 Module Documentation

9.3 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.

9.3.1 Detailed Description

9.4 SCU firmware selection 23

9.4 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.

9.4.1 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.

24 Module Documentation

9.5 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.

9.5.1 Detailed Description

CANopen function codes defined in DS301 profile.

9.6 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

9.6.1 Detailed Description

26 Module Documentation

9.7 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.

9.7.1 Detailed Description

9.7.2 Function Documentation

9.7.2.1 filter_buffer()

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	

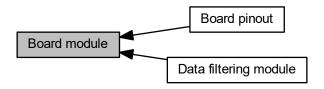
Returns

Filtered data

Definition at line 39 of file filter.cpp.

9.8 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.

9.8 Board module 29

#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.

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_SUSP_ADC_CHAN_NUM | RT_DX_SUSP_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)

9.8 Board module 31

```
• 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 uint16_t fr_sx_rpm

      Frontal left wheel velocity [ rpm].

    volatile uint16 t fr dx rpm

      Frontal right wheel velocity [ rpm].
• 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].

9.8.1 Detailed Description

9.8.2 Macro Definition Documentation

```
9.8.2.1 BUFFERS
```

```
#define BUFFERS 4
```

Number of DMA buffers.

Warning

Must be a power of two

Definition at line 30 of file model.cpp.

9.8.3 Function Documentation

```
9.8.3.1 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 667 of file model.cpp.

```
9.8.3.2 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.uniromal.it)
```

Definition at line 483 of file model.cpp.

9.8 Board module 33

```
9.8.3.3 fr_sx_pulse()

void fr_sx_pulse ( )

EXTI IRQ handler. Extended.
```

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 470 of file model.cpp.

```
9.8.3.4 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.uniromal.it)
```

Returns

Frontal right wheel rpm

Definition at line 512 of file model.cpp.

```
9.8.3.5 get_fr_sx_rpm()
volatile uint16_t get_fr_sx_rpm ( )
```

If rpm value is lower than $\ensuremath{\mathsf{RPM_MIN}}$, output is forced to zero.

This function returns frontal left wheel velocity [rpm].

Author

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

Returns

Frontal left wheel rpm

Definition at line 497 of file model.cpp.

```
9.8.3.6 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.uniromal.it)
Returns
     Retro right wheel rpm
9.8.3.7 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
9.8.3.8 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.uniromal.it)
```

Definition at line 700 of file model.cpp.

9.8 Board module 35

9.8.4 Variable Documentation

```
9.8.4.1 apps_plausibility [1/2]
volatile bool apps_plausibility
APPS plausibility status.
APPS plausibility flag.
Definition at line 338 of file model.cpp.
9.8.4.2 apps_plausibility [2/2]
volatile bool apps_plausibility = true
APPS plausibility status.
APPS plausibility flag.
Definition at line 338 of file model.cpp.
9.8.4.3 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.
9.8.4.4 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.

```
9.8.4.5 brake_plausibility [1/2]
volatile bool brake_plausibility
Brake plausibility status.
Brake plausibility flag.
Definition at line 344 of file model.cpp.
9.8.4.6 brake_plausibility [2/2]
volatile uint8_t brake_plausibility = true
Brake plausibility status.
Brake plausibility flag.
Definition at line 344 of file model.cpp.
9.8.4.7 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.
9.8.4.8 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.
9.8.4.9 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.
9.8.4.10 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.

9.9 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.

9.9.1 Detailed Description

9.9.2 Function Documentation

9.9.2.1 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.

```
9.9.2.2 slaveSendBootUp()
```

```
void slaveSendBootUp ( )
```

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

Author

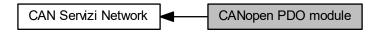
Arella Matteo

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

Definition at line 59 of file nmt.cpp.

9.10 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.

9.10.1 Detailed Description

Data into PDOs are configured according TPDO_configuration.

9.10.2 Function Documentation

9.10.2.1 buildPDO()

This function serializes data to send into PDO message.

Author

Arella Matteo

```
(mail: arella.1646983@studenti.uniromal.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.

```
9.10.2.2 proceedPDO()
```

This function manages PDO message receive, deserializing data.

Author

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

Parameters

in	pdo	CANopen message to manage
----	-----	---------------------------

Definition at line 56 of file pdo.cpp.

9.11 Radio module 41

9.11 Radio module

Macros

· #define CTR 1

Enable/Disable aes CTR mode.

• #define RADIO_KEY_BITS 192

Aes key length [bits].

Size of static buffer for JSON serialization, generated by https://arduinojson.org/assistant/. Model serialization format:

• #define CIPHER MAX LENGTH 1024

Cipher buffer max length (according to https://arduinojson.org/assistant/).

#define IV_LEN AES_KEYLEN

Initialization Vector length.

Functions

RF24 radio (RADIO_CE_PIN, RADIO_CSN_PIN)

Radio

• void encrypt_model (char *buffer, char *iv, uint16_t plain_len, uint16_t buffer_len)

Encrypt model for transmit over radio.

• volatile char generate_random_char ()

Generate a Cryptgraphically secure pseudorandom number from TRNG hardware peripheral.

void generate_iv (char *buffer, uint16_t len)

Generate a randomized initialization vector.

void pkcs7_padding (char *buffer, uint16_t plain_len, uint16_t buffer_len)

Perform PKCS7 padding described in RFC 5652.

• void byte padding (char *buffer, uint16 t plain len, uint16 t buffer len)

Perform ISO/IEC 7816-4 byte padding.

• void radio init ()

Initialize TRNG (True Random Number Generator) hardware peripheral as a CSPRNG (Cryptographically Secure Pseudo-Random Number Generator) for generate randomized IV.

• void radio_send_model ()

Actions performed involve:

Variables

• volatile bool radio_transmit = false

Radio transmit enable flag.

char key [AES_KEYLEN]

AES encryption KEY.

• char iv [IV_LEN+1]

AES encryption Initialization Vector.

• char cipher [CIPHER_MAX_LENGTH+1] = {0}

Encrypted model buffer.

const uint64_t pipe = 0xE8E8F0F0E1LL

Radio writing pipe.

· volatile bool radio_transmit

Radio transmit enable flag.

9.11.1 Detailed Description

9.11.2 Macro Definition Documentation

9.11.2.1 CIPHER MAX LENGTH

```
#define CIPHER_MAX_LENGTH 1024
```

Cipher buffer max length (according to https://arduinojson.org/assistant/).

Warning

Must be a multiple of 16.

Definition at line 64 of file radio.cpp.

9.11.2.2 JSON BUFFER SIZE

```
#define JSON_BUFFER_SIZE JSON_OBJECT_SIZE(2) + 3 * JSON_OBJECT_SIZE(4) + JSON_OBJECT_SIZE(5) +
219
```

Size of static buffer for JSON serialization, generated by https://arduinojson.org/assistant/. Model serialization format:

```
{"pedals":{"tps1":XX,"tps2":XX,"brake":XX,"apps_plaus":true,"brake_plaus":true},
   "suspensions":{"front_sx":XX,"front_dx":XX,"retro_sx":XX,"retro_dx":XX},
   "wheels":{"front_sx":XXXX,"front_dx":XXXX,"retro_sx":XXXX,"retro_dx":XXXX},
   "accelerometers":{"acc_x":X,"acc_z":X}}
```

Definition at line 57 of file radio.cpp.

9.11.3 Function Documentation

9.11.3.1 byte_padding()

Perform ISO/IEC 7816-4 byte padding.

Author

```
Arella Matteo
```

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

9.11 Radio module 43

Parameters

out	buffer	- Buffer
in	plain_len	- Plain model's length
in	buffer_len	- Buffer length

Definition at line 185 of file radio.cpp.

9.11.3.2 encrypt_model()

Encrypt model for transmit over radio.

Author

Arella Matteo

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

Parameters

in,out	buffer	- Buffer to be encrypted
in	iv	- Initialisation vector for AES encryption
in	plain_len	- Plain model's length
in	buffer_len	- Buffer length

Definition at line 193 of file radio.cpp.

9.11.3.3 generate_iv()

Generate a randomized initialization vector.

Author

Arella Matteo

```
(mail: arella.1646983@studenti.uniromal.it)
```

Parameters

out	buffer	- Initialisation Vector
in	len	- Buffer length

Definition at line 152 of file radio.cpp.

```
9.11.3.4 generate_random_char()
```

```
volatile char generate_random_char ( )
```

Generate a Cryptgraphically secure pseudorandom number from TRNG hardware peripheral.

Author

```
Arella Matteo
```

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

Returns

Cryptgraphically secure pseudorandom number.

Definition at line 138 of file radio.cpp.

9.11.3.5 pkcs7_padding()

Perform PKCS7 padding described in RFC 5652.

Author

Arella Mattec

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

Parameters

out	buffer	- Buffer
in	plain_len	- Plain model's length
in	buffer_len	- Buffer length

9.11 Radio module 45

Definition at line 169 of file radio.cpp.

```
9.11.3.6 radio_init()
void radio_init ( )
```

Initialize TRNG (True Random Number Generator) hardware peripheral as a CSPRNG (Cryptographically Secure Pseudo-Random Number Generator) for generate randomized IV.

Initialize radio.

Author

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

Definition at line 213 of file radio.cpp.

```
9.11.3.7 radio_send_model()

void radio_send_model ( )
```

Actions performed involve:

Send vehicle model over radio.

- · serialize vehicle data into static JSON buffer;
- · generate randomised IV;
- add padding to buffer;
- encrypt model with 192-bit AES encryption (CTR mode);
- · encode buffer with base64 encoding;
- · send buffer over radio.

Author

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

Definition at line 236 of file radio.cpp.

9.11.4 Variable Documentation

```
9.11.4.1 iv

char iv[IV_LEN+1]

AES encryption Initialization Vector.
```

Warning

IV must never be reused with the same key

Definition at line 95 of file radio.cpp.

9.12 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.

9.12.1 Detailed Description

9.12.2 Function Documentation

```
9.12.2.1 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.uniromal.it)
```

Definition at line 79 of file SCU.ino.

```
9.12.2.2 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.uniromal.it)
```

Definition at line 59 of file SCU.ino.

9.13 Board pinout 47

9.13 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_SUSP_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_SUSP_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.

9.13.1 Detailed Description

Board pinout for each sensor, CAN port and radio.

9.14 CANopen Finite State Machine module

Collaboration diagram for CANopen Finite State Machine module:



Typedefs

• typedef enum enum_nodeState e_nodeState

Enumerations

```
    enum enum_nodeState {
    Initialisation = 0x00, Disconnected = 0x01, Connecting = 0x02, Preparing = 0x02,
    Stopped = 0x04, Operational = 0x05, Pre operational = 0x7F, Unknown state = 0x0F }
```

Functions

• e_nodeState getState ()

Return current state on the Finite State Machine (FSM).

void setState (e_nodeState newState)

Set current state on the Finite State Machine (FSM).

void canDispatch (Message *m)

Called by driver when receiving CANopen messages.

• void initialisation ()

Initialize Board module. If rear SCU firmware is selected (according to SCU firmware selection) radio is initialized. It initializes the entire application, CAN/CANopen interfaces and communication.

• void preOperational ()

preOperational state task on the FSM.

• void operational ()

Start timer for periodic TPDO transmit according to CAN Servizi network.

void stopped ()

Stop timer for periodic TPDO transmit according to CAN Servizi network.

Variables

• volatile e_nodeState current_state = Initialisation

Current state of FSM.

9.14.1 Detailed Description

9.14.2 Typedef Documentation

9.14.2.1 e_nodeState

```
typedef enum enum_nodeState e_nodeState
```

FSM states typedef

Definition at line 55 of file states.h.

9.14.3 Enumeration Type Documentation

9.14.3.1 enum_nodeState

```
enum enum_nodeState
```

FSM states enum

Enumerator

Initialisation	Initialisation state
Disconnected	Disconnected state
Connecting	Connecting state
Preparing	Preparing state
Stopped	Stopped state
Operational	Operational state
Pre_operational	PreOperational state
Unknown_state	Unknown state

Definition at line 43 of file states.h.

9.14.4 Function Documentation

9.14.4.1 canDispatch()

```
void can
Dispatch ( {\tt Message} \ * \ {\tt m} \ )
```

Called by driver when receiving CANopen messages.

Author

Arella Matteo

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

Parameters

```
in m Received CANopen message
```

Definition at line 43 of file states.cpp.

```
9.14.4.2 getState()
```

```
e_nodeState getState ( )
```

Return current state on the Finite State Machine (FSM).

Author

Arella Matteo

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

Returns

current state on the FSM

Definition at line 34 of file states.cpp.

9.14.4.3 initialisation()

```
void initialisation ( )
```

Initialize Board module. If rear SCU firmware is selected (according to SCU firmware selection) radio is initialized. It initializes the entire application, CAN/CANopen interfaces and communication.

Initialisation state task on the FSM.

Author

```
Arella Matteo
```

```
(mail: arella.1646983@studenti.uniromal.it)
```

Definition at line 69 of file states.cpp.

```
9.14.4.4 operational()
void operational ( )
Start timer for periodic TPDO transmit according to CAN Servizi network.
Operational state task on the FSM.
Author
     Arella Matteo
     (mail: arella.1646983@studenti.uniroma1.it)
Definition at line 91 of file states.cpp.
9.14.4.5 preOperational()
void preOperational ( )
preOperational state task on the FSM.
Author
     Arella Matteo
     (mail: arella.1646983@studenti.uniroma1.it)
Definition at line 80 of file states.cpp.
9.14.4.6 setState()
void setState (
              e_nodeState newState )
Set current state on the Finite State Machine (FSM).
Author
     Arella Matteo
     (mail: arella.1646983@studenti.uniroma1.it)
```

Parameters

in *newState* New state transition

Definition at line 38 of file states.cpp.

```
9.14.4.7 stopped()

void stopped ( )

Stop timer for periodic TPDO transmit according to CAN Servizi network.

Stopped state task on the FSM.

Author

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

Definition at line 103 of file states.cpp.

9.15 CANopen Timer module

Collaboration diagram for CANopen Timer module:



Macros

#define SCU FRONT FIRST SLOT 0

First time slot for $SCU_{Frontal}$.

• #define SCU_FRONT_SECOND_SLOT 1

Second time slot for $SCU_{Frontal}$.

• #define SCU REAR SLOT 2

Time slot for SCU_{Rear} .

• #define TCS_SLOT 3

Time slot for TCU.

• #define TIME_SLOT_MASK 3

Time slot mask.

• #define RADIO_SLOT_MASK 7

Radio submit slot mask: number of SCU_{Rear} time slots between one submit and successive one (TIME_SLOT_PERIOD * RADIO_SLOT_MASK * num. time slots between previous and current SCU_{Rear})

Functions

void TimeDispatch ()

Dispatch time slot for each CANopen node according to TPDO_Timer.

• void timerInit ()

Initialize timer for periodic submit of TPDOs.

· void timerStart ()

Start timer for periodic submit of TPDOs.

• void timerStop ()

Stop timer for periodic submit of TPDOs.

Variables

volatile uint8_t t_slot = SCU_FRONT_FIRST_SLOT

Current time slot.

• volatile uint8_t radio_slot = 0

Current radio time slot.

• DueTimer * timer

Timer for periodic TPDO submit.

9.15.1 Detailed Description

9.15.2 Function Documentation

```
9.15.2.1 TimeDispatch()
void TimeDispatch ( )
Dispatch time slot for each CANopen node according to TPDO_Timer.
Author
     Arella Matteo
     (mail: arella.1646983@studenti.uniroma1.it)
Definition at line 83 of file timer.cpp.
9.15.2.2 timerInit()
void timerInit ( )
Initialize timer for periodic submit of TPDOs.
Author
     Arella Matteo
     (mail: arella.1646983@studenti.uniroma1.it)
Definition at line 118 of file timer.cpp.
9.15.2.3 timerStart()
void timerStart ( )
Start timer for periodic submit of TPDOs.
Author
     Arella Matteo
     (mail: arella.1646983@studenti.uniroma1.it)
Definition at line 122 of file timer.cpp.
9.15.2.4 timerStop()
void timerStop ( )
Stop timer for periodic submit of TPDOs.
Author
     Arella Matteo
     (mail: arella.1646983@studenti.uniroma1.it)
```

Definition at line 126 of file timer.cpp.

Chapter 10

Class Documentation

10.1 Message Struct Reference

CANopen message struct.

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

Public Attributes

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

10.1.1 Detailed Description

CANopen message struct.

Definition at line 91 of file CO_can.h.

10.1.2 Member Data Documentation

10.1.2.1 cob_id

uint16_t Message::cob_id

message's COB-ID

Definition at line 92 of file CO_can.h.

56 Class Documentation

10.1.2.2 data

uint8_t Message::data[8]

message's datas

Definition at line 94 of file CO_can.h.

10.1.2.3 len

uint8_t Message::len

message's length (0 to 8)

Definition at line 93 of file CO_can.h.

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

• CO_can.h

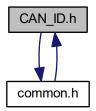
Chapter 11

File Documentation

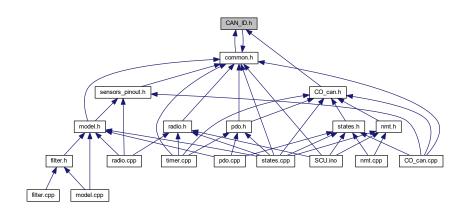
11.1 CAN_ID.h File Reference

CAN servizi nodeIDs module header.

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



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



58 File Documentation

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.

11.1.1 Detailed Description

CAN servizi nodelDs module header.

Author

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

Date

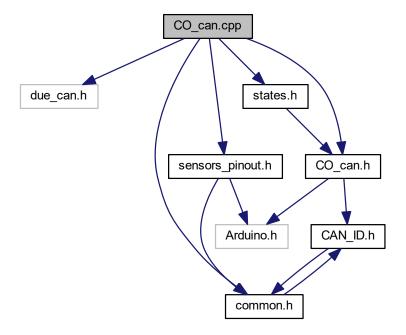
2018

11.2 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 "sensors_pinout.h"
```

Include dependency graph for CO_can.cpp:



Functions

• uint8_t getNodeld ()

This function returns node ID into CAN servizi network.

void setNodeld (uint8_t nodeld)

This function sets node ID into CAN servizi network.

void canSend (Message *m)

This function send a CANopen message over CAN servizi network.

void CAN_general_callback (CAN_FRAME *frame)

This function manage CAN frame receiption and dispatch CANopen message.

• void initCAN ()

This function initialize CAN/CANopen interfaces and communication.

Variables

• uint8_t nodeld

Board node ID into CAN servizi network according to NODE-ID.

11.2.1 Detailed Description

CANOpen main module implementation file.

60 File Documentation

Author

Arella Matteo

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

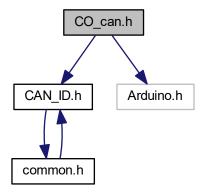
Date

2018

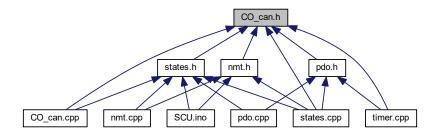
11.3 CO_can.h File Reference

CANOpen main module header.

```
#include "CAN_ID.h"
#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.

• uint8_t getNodeld ()

This function returns node ID into CAN servizi network.

void setNodeld (uint8_t nodeld)

This function sets node ID into CAN servizi network.

11.3.1 Detailed Description

CANOpen main module header.

Author

```
Arella Matteo
```

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

Date

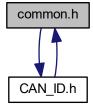
2018

11.4 common.h File Reference

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

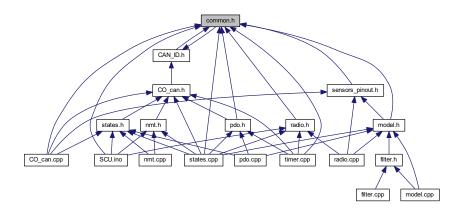
```
#include "CAN_ID.h"
Include dependency graph for common.h:
```

, , , ,



62 File Documentation

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.

11.4.1 Detailed Description

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

Author

```
Arella Matteo
```

(mail: arella.1646983@studenti.uniromal.it)

Date

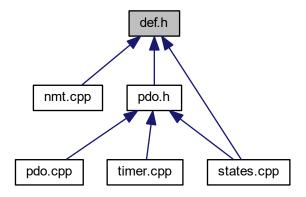
2018

11.5 def.h File Reference 63

11.5 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

11.5.1 Detailed Description

CANopen DS301 definitions.

Author

Arella Matteo

(mail: arella.1646983@studenti.uniromal.it)

Date

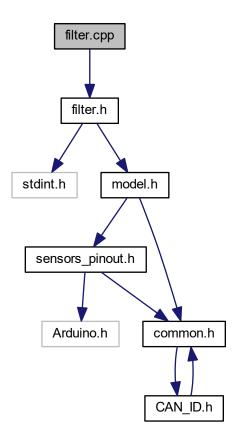
2018

11.6 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.

11.6.1 Detailed Description

Filter module implementation file.

Author

Arella Matteo

(mail: arella.1646983@studenti.uniromal.it)

Date

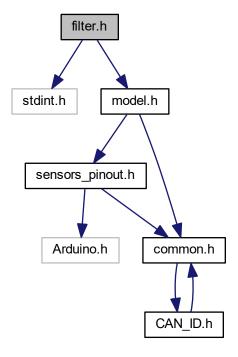
2018

11.7 filter.h File Reference

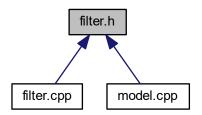
Filter module header file.

#include <stdint.h>
#include "model.h"
look decorded as graph for filter.

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.

11.7.1 Detailed Description

Filter module header file.

Author

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

Date

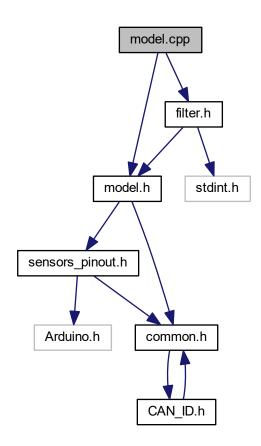
2018

11.8 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_SUSP_ADC_CHAN_NUM | FR_DX_SUSP_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_SUSP_ADC_CHAN_NUM | RT_DX_SUSP_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.

11.8.1 Detailed Description

Board model implementation file.

Author

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

Date

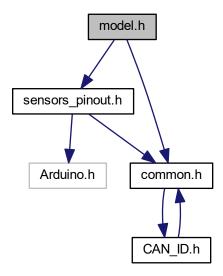
2018

11.9 model.h File Reference

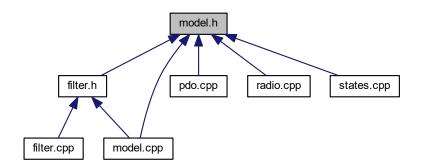
Board model header file.

```
#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 uint16_t fr_sx_rpm

           Frontal left wheel velocity [ rpm].

    volatile uint16_t fr_dx_rpm

           Frontal right wheel velocity [ rpm].
    • 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].
11.9.1
         Detailed Description
Board model header file.
Author
      Arella Matteo
      (mail: arella.1646983@studenti.uniroma1.it)
Date
```

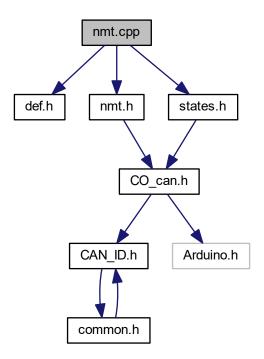
Generated by Doxygen

11.10 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.

11.10.1 Detailed Description

CANOpen NMT module implementation file.

Author

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

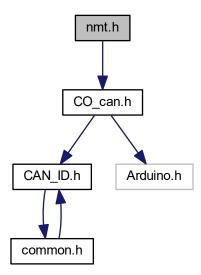
Date

11.11 nmt.h File Reference 75

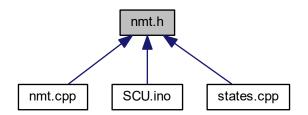
11.11 nmt.h File Reference

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.

11.11.1 Detailed Description

CANOpen NMT module header.

Author

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

Date

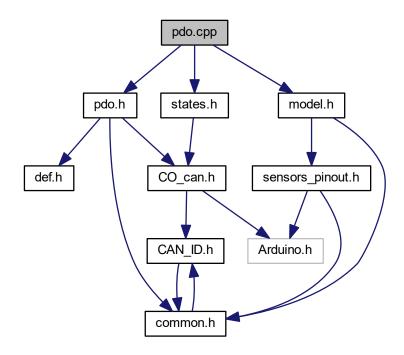
2018

11.12 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.

11.12.1 Detailed Description

CANopen PDO support header file.

Author

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

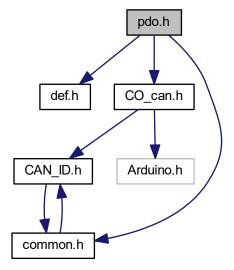
Date

2018

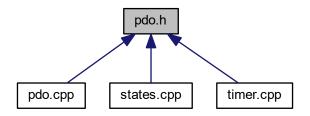
11.13 pdo.h File Reference

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.

11.13.1 Detailed Description

CANopen PDO support header file.

Author

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

Date

2018

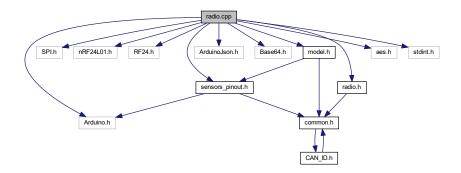
11.14 radio.cpp File Reference

Radio implementation file.

```
#include "radio.h"
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Arduino.h>
#include <ArduinoJson.h>
#include <Base64.h>
#include "sensors_pinout.h"
#include "model.h"
```

#include "aes.h"

Include dependency graph for radio.cpp:



Macros

• #define CTR 1

Enable/Disable aes CTR mode.

• #define RADIO_KEY_BITS 192

Aes key length [bits].

Size of static buffer for JSON serialization, generated by https://arduinojson.org/assistant/. Model serialization format:

#define CIPHER_MAX_LENGTH 1024

Cipher buffer max length (according to https://arduinojson.org/assistant/).

• #define IV LEN AES KEYLEN

Initialization Vector length.

Functions

RF24 radio (RADIO_CE_PIN, RADIO_CSN_PIN)

Radio.

• void encrypt_model (char *buffer, char *iv, uint16_t plain_len, uint16_t buffer_len)

Encrypt model for transmit over radio.

• volatile char generate_random_char ()

Generate a Cryptgraphically secure pseudorandom number from TRNG hardware peripheral.

• void generate_iv (char *buffer, uint16_t len)

Generate a randomized initialization vector.

• void pkcs7_padding (char *buffer, uint16_t plain_len, uint16_t buffer_len)

Perform PKCS7 padding described in RFC 5652.

• void byte_padding (char *buffer, uint16_t plain_len, uint16_t buffer_len)

Perform ISO/IEC 7816-4 byte padding.

• void radio_init ()

Initialize TRNG (True Random Number Generator) hardware peripheral as a CSPRNG (Cryptographically Secure Pseudo-Random Number Generator) for generate randomized IV.

· void radio_send_model ()

Actions performed involve:

Variables

```
• volatile bool radio_transmit = false
```

Radio transmit enable flag.

• char key [AES_KEYLEN]

AES encryption KEY.

• char iv [IV_LEN+1]

AES encryption Initialization Vector.

• char cipher [CIPHER_MAX_LENGTH+1] = {0}

Encrypted model buffer.

• const uint64_t pipe = 0xE8E8F0F0E1LL

Radio writing pipe.

11.14.1 Detailed Description

Radio implementation file.

Author

```
Arella Matteo
```

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

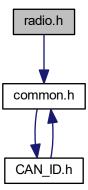
Date

2018

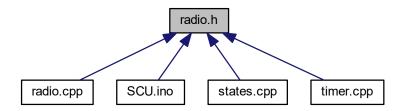
11.15 radio.h File Reference

Radio header file.

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



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



Functions

• void radio_init ()

Initialize TRNG (True Random Number Generator) hardware peripheral as a CSPRNG (Cryptographically Secure Pseudo-Random Number Generator) for generate randomized IV.

• void radio_send_model ()

Actions performed involve:

Variables

· volatile bool radio_transmit

Radio transmit enable flag.

11.15.1 Detailed Description

Radio header file.

Author

```
Arella Matteo
```

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

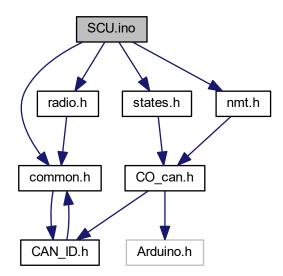
Date

11.16 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.

11.16.1 Detailed Description

Main module file.

Author

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

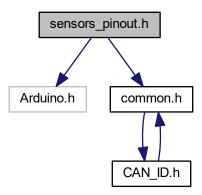
Date

11.17 sensors_pinout.h File Reference

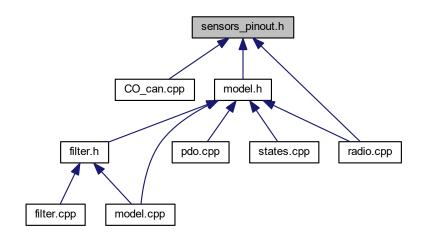
Board pinout module header.

```
#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_SUSP_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_SUSP_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.

11.17.1 Detailed Description

Board pinout module header.

Author

```
Arella Matteo
```

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

Date

2018

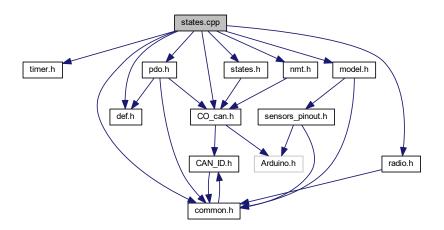
11.18 states.cpp File Reference

CANopen finite state machine implementation file.

```
#include "timer.h"
#include "def.h"
#include "pdo.h"
#include "states.h"
#include "nmt.h"
#include "CO_can.h"
#include "common.h"
```

```
#include "model.h"
#include "radio.h"
```

Include dependency graph for states.cpp:



Functions

• e nodeState getState ()

Return current state on the Finite State Machine (FSM).

void setState (e_nodeState newState)

Set current state on the Finite State Machine (FSM).

void canDispatch (Message *m)

Called by driver when receiving CANopen messages.

· void initialisation ()

Initialize Board module. If rear SCU firmware is selected (according to SCU firmware selection) radio is initialized. It initializes the entire application, CAN/CANopen interfaces and communication.

• void preOperational ()

preOperational state task on the FSM.

· void operational ()

Start timer for periodic TPDO transmit according to CAN Servizi network.

· void stopped ()

Stop timer for periodic TPDO transmit according to CAN Servizi network.

Variables

volatile e_nodeState current_state = Initialisation
 Current state of FSM.

11.18.1 Detailed Description

CANopen finite state machine implementation file.

Author

Arella Matteo

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

Date

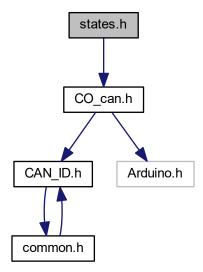
2018

11.19 states.h File Reference

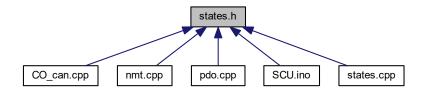
CANopen finite state machine header file.

#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

Enumerations

```
    enum enum_nodeState {
    Initialisation = 0x00, Disconnected = 0x01, Connecting = 0x02, Preparing = 0x02,
    Stopped = 0x04, Operational = 0x05, Pre_operational = 0x7F, Unknown_state = 0x0F }
```

Functions

· void initialisation ()

Initialize Board module. If rear SCU firmware is selected (according to SCU firmware selection) radio is initialized. It initializes the entire application, CAN/CANopen interfaces and communication.

void preOperational ()

preOperational state task on the FSM.

• void operational ()

Start timer for periodic TPDO transmit according to CAN Servizi network.

void stopped ()

Stop timer for periodic TPDO transmit according to CAN Servizi network.

void canDispatch (Message *m)

Called by driver when receiving CANopen messages.

• e_nodeState getState ()

Return current state on the Finite State Machine (FSM).

void setState (e_nodeState newState)

Set current state on the Finite State Machine (FSM).

11.19.1 Detailed Description

CANopen finite state machine header file.

Author

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

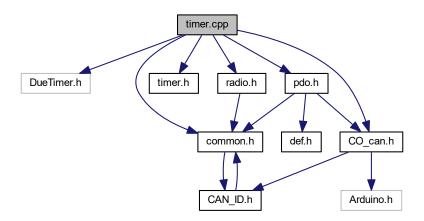
Date

11.20 timer.cpp File Reference

CANopen timer implementation file.

```
#include <DueTimer.h>
#include "common.h"
#include "timer.h"
#include "pdo.h"
#include "CO_can.h"
#include "radio.h"
```

Include dependency graph for timer.cpp:



Macros

• #define SCU_FRONT_FIRST_SLOT 0

First time slot for $SCU_{Frontal}$.

• #define SCU_FRONT_SECOND_SLOT 1

Second time slot for $SCU_{Frontal}$.

• #define SCU_REAR_SLOT 2

Time slot for SCU_{Rear} .

• #define TCS_SLOT 3

Time slot for TCU.

• #define TIME_SLOT_MASK 3

Time slot mask.

#define RADIO_SLOT_MASK 7

Radio submit slot mask: number of SCU_{Rear} time slots between one submit and successive one (TIME_SLOT_PERIOD * RADIO_SLOT_MASK * num. time slots between previous and current SCU_{Rear})

Functions

· void TimeDispatch ()

Dispatch time slot for each CANopen node according to TPDO_Timer.

void timerInit ()

11.21 timer.h File Reference

89

Initialize timer for periodic submit of TPDOs.

• void timerStart ()

Start timer for periodic submit of TPDOs.

• void timerStop ()

Stop timer for periodic submit of TPDOs.

Variables

```
• volatile uint8_t t_slot = SCU_FRONT_FIRST_SLOT
```

Current time slot.

• volatile uint8_t radio_slot = 0

Current radio time slot.

• DueTimer * timer

Timer for periodic TPDO submit.

11.20.1 Detailed Description

CANopen timer implementation file.

Author

```
Arella Matteo
```

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

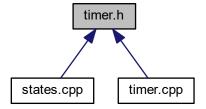
Date

2018

11.21 timer.h File Reference

CANopen timer header file.

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



Functions

```
• void timerInit ()
```

Initialize timer for periodic submit of TPDOs.

• void timerStart ()

Start timer for periodic submit of TPDOs.

• void timerStop ()

Stop timer for periodic submit of TPDOs.

• void TimeDispatch ()

Dispatch time slot for each CANopen node according to TPDO_Timer.

11.21.1 Detailed Description

CANopen timer header file.

Author

```
Arella Matteo
```

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

Date

Index

ADC Handler	setState, 51
Board module, 32	stopped, 52
apps_plausibility	CANopen Function Codes, 24
Board module, 35	CANopen NMT Command Specifiers, 25
Board Moddie, 00	CANopen NMT module, 37
BUFFERS	proceedNMTstateChange, 37
Board module, 32	slaveSendBootUp, 38
Board module, 28	CANopen PDO module, 39
ADC_Handler, 32	buildPDO, 39
apps_plausibility, 35	proceedPDO, 40
BUFFERS, 32	CANopen Timer module, 53
brake_percentage, 35	TimeDispatch, 54
brake_plausibility, 35, 36	timerlnit, 54
fr_dx_pulse, 32	timerStart, 54
fr_sx_pulse, 32	timerStop, 54
get_fr_dx_rpm, 33	CIPHER_MAX_LENGTH
get_fr_sx_rpm, 33	Radio module, 42
get_rt_dx_rpm, 33	CO_can.cpp, 58
get_rt_sx_rpm, 34	CO_can.h, 60
model_init, 34	canDispatch
tps1_percentage, 36	CANopen Finite State Machine module, 49
tps2_percentage, 36	canSend
Board pinout, 47	CAN Servizi Network, 19
brake percentage	cob_id
Board module, 35	Message, 55
brake_plausibility	Common Defines, 22
Board module, 35, 36	common.h, 61
buildPDO	
CANopen PDO module, 39	data
byte_padding	Message, 55
Radio module, 42	Data filtering module, 26
,	filter_buffer, 26
CAN Network Nodes ID, 21	def.h, 63
CAN Servizi Network, 17	
CAN_general_callback, 18	e_nodeState
canSend, 19	CANopen Finite State Machine module, 49
getNodeld, 19	encrypt_model
initCAN, 19	Radio module, 43
setNodeld, 20	enum_nodeState
CAN_ID.h, 57	CANopen Finite State Machine module, 49
CAN general callback	filter.cpp, 64
CAN Servizi Network, 18	filter.h, 66
CANopen Finite State Machine module, 48	filter buffer
canDispatch, 49	Data filtering module, 26
e_nodeState, 49	fr_dx_pulse
enum_nodeState, 49	Board module, 32
getState, 50	
initialisation, 50	fr_sx_pulse Board module, 32
operational, 50	Dogia module, 32
preOperational, 51	generate_iv
· · · · · · · · · · · · · · · · · · ·	- — — — — — — — — — — — — — — — — — — —

92 INDEX

Radio module, 43	Radio module, 41
generate random char	byte padding, 42
Radio module, 44	CIPHER MAX LENGTH, 42
get_fr_dx_rpm	encrypt_model, 43
Board module, 33	generate_iv, 43
get_fr_sx_rpm	generate_random_char, 44
Board module, 33	iv, 45
get_rt_dx_rpm	JSON_BUFFER_SIZE, 42
Board module, 33	pkcs7_padding, 44
get_rt_sx_rpm	radio_init, 45
Board module, 34	radio_send_model, 45
getNodeld	radio.cpp, 78
CAN Servizi Network, 19	radio.h, 80
getState	radio init
CANopen Finite State Machine module, 50	Radio module, 45
or tropon i mito otato maorinio modalo, oo	radio_send_model
initCAN	Radio module, 45
CAN Servizi Network, 19	ridate friedate, re
initialisation	SCU firmware selection, 23
CANopen Finite State Machine module, 50	SCU.ino, 82
iv	sensors_pinout.h, 83
Radio module, 45	setNodeld
Hadio Module, 40	CAN Servizi Network, 20
JSON_BUFFER_SIZE	setState
Radio module, 42	CANopen Finite State Machine module, 51
hadio module, 42	setup
len	Main module, 46
	slaveSendBootUp
Message, 56	CANopen NMT module, 38
loop Main madula 46	states.cpp, 84
Main module, 46	states.h, 86
Main modulo, 46	
Main module, 46	stopped CANopen Finite State Machine module, 52
loop, 46	CANopen Finite State Machine module, 52
setup, 46	TimeDispatch
Message, 55	CANopen Timer module, 54
cob_id, 55	timer.cpp, 88
data, 55	timer.h, 89
len, 56	timerinit
model.cpp, 67	CANopen Timer module, 54
model.h, 71	timerStart
model_init	
Board module, 34	CANopen Timer module, 54
. 74	timerStop
nmt.cpp, 74	CANopen Timer module, 54
nmt.h, 75	tps1_percentage
	Board module, 36
operational	tps2_percentage
CANopen Finite State Machine module, 50	Board module, 36
pdo.cpp, 76	
pdo.h, 77	
pkcs7_padding	
Radio module, 44	
preOperational	
CANopen Finite State Machine module, 51	
proceedNMTstateChange	
CANopen NMT module, 37	
proceedPDO	
CANopen PDO module, 40	