

## Ethernet Reader

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	AvgStruct Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Member Data Documentation . . . . .	5
3.1.2.1	samples . . . . .	5
3.1.2.2	sum . . . . .	5
3.2	CommStruct Struct Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.2.2	Member Data Documentation . . . . .	6
3.2.2.1	addr_dest . . . . .	6
3.2.2.2	answer_id . . . . .	6
3.2.2.3	answer_timeout_flag . . . . .	6
3.2.2.4	answer_tmr . . . . .	7
3.2.2.5	awaiting_answer . . . . .	7
3.2.2.6	send_tries . . . . .	7
3.2.2.7	sent_request . . . . .	7
3.2.2.8	source_dev . . . . .	7
3.3	CommTestResultStruct Struct Reference . . . . .	7

3.3.1	Detailed Description	8
3.3.2	Member Data Documentation	8
3.3.2.1	completed	8
3.3.2.2	in_progress	8
3.3.2.3	nok	8
3.3.2.4	ok	8
3.3.2.5	reserved	8
3.4	CommTestSettStruct Struct Reference	9
3.4.1	Detailed Description	9
3.4.2	Member Data Documentation	9
3.4.2.1	addr	9
3.4.2.2	cnt	9
3.4.2.3	data_len	9
3.4.2.4	timeout	9
3.5	EthernetReader_Cfg Struct Reference	10
3.5.1	Detailed Description	10
3.5.2	Member Data Documentation	10
3.5.2.1	header	10
3.5.2.2	module1_on	10
3.5.2.3	module2_on	10
3.5.2.4	nanotron_clk_en	11
3.5.2.5	reserved	11
3.5.2.6	route_from_wireless1	11
3.5.2.7	route_from_wireless2	11
3.5.2.8	usb_enabled	11
3.6	EthernetReader_DW_Cfg Struct Reference	11
3.6.1	Detailed Description	12
3.6.2	Member Data Documentation	12
3.6.2.1	header	12
3.7	EthernetReader_NTR_Cfg Struct Reference	12

3.7.1	Detailed Description	12
3.7.2	Member Data Documentation	12
3.7.2.1	header	13
3.7.2.2	mode	13
3.7.2.3	nano_pwr	13
3.8	EthernetReader_State Struct Reference	13
3.8.1	Detailed Description	14
3.8.2	Member Data Documentation	14
3.8.2.1	batt_voltage	14
3.8.2.2	charging	14
3.8.2.3	chrg_current	14
3.8.2.4	chrg_mode	14
3.8.2.5	header	14
3.8.2.6	module1_rssi_back	14
3.8.2.7	module1_type	15
3.8.2.8	module2_rssi_back	15
3.8.2.9	module2_type	15
3.8.2.10	pwr_good	15
3.8.2.11	reserved	15
3.8.2.12	server_connected	15
3.8.2.13	supply_voltage	15
3.8.2.14	temperature	15
3.8.2.15	uptime	16
3.8.2.16	usb_voltage	16
3.9	EthernetReader_SX_Cfg Struct Reference	16
3.9.1	Detailed Description	16
3.9.2	Member Data Documentation	16
3.9.2.1	clk_out	16
3.9.2.2	header	17
3.9.2.3	if_gain	17

3.9.2.4	<a href="#">listen_on_chn2</a>	17
3.9.2.5	<a href="#">reserved</a>	17
3.9.2.6	<a href="#">rf_frequency_1</a>	17
3.9.2.7	<a href="#">rf_frequency_2</a>	17
3.9.2.8	<a href="#">rf_power</a>	17
3.9.2.9	<a href="#">vco_trim</a>	18
3.10	<a href="#">Flag_struct Struct Reference</a>	18
3.10.1	<a href="#">Detailed Description</a>	18
3.10.2	<a href="#">Member Data Documentation</a>	18
3.10.2.1	<a href="#">ms100</a>	18
3.10.2.2	<a href="#">power_off</a>	18
3.10.2.3	<a href="#">reinit_modules</a>	19
3.10.2.4	<a href="#">sec</a>	19
3.10.2.5	<a href="#">sx1_irq_detected</a>	19
3.10.2.6	<a href="#">sx2_irq_detected</a>	19
3.10.2.7	<a href="#">update_cfg</a>	19
3.10.2.8	<a href="#">usb_forced</a>	19
3.10.2.9	<a href="#">wakeup_burst</a>	19
3.11	<a href="#">Leds_struct Struct Reference</a>	20
3.11.1	<a href="#">Detailed Description</a>	20
3.11.2	<a href="#">Member Data Documentation</a>	20
3.11.2.1	<a href="#">conn</a>	20
3.11.2.2	<a href="#">conn_blink</a>	20
3.11.2.3	<a href="#">err</a>	20
3.11.2.4	<a href="#">mod1</a>	21
3.11.2.5	<a href="#">mod1_blink</a>	21
3.11.2.6	<a href="#">mod2</a>	21
3.11.2.7	<a href="#">mod2_blink</a>	21
3.11.2.8	<a href="#">pwr</a>	21
3.11.2.9	<a href="#">reserved</a>	21

3.11.2.10 sys . . . . .	21
3.12 MODULE_STRUCT Struct Reference . . . . .	22
3.12.1 Detailed Description . . . . .	22
3.12.2 Member Data Documentation . . . . .	22
3.12.2.1 slot1_type . . . . .	22
3.12.2.2 slot2_type . . . . .	22
3.13 Module_SX_Cfg Struct Reference . . . . .	22
3.13.1 Detailed Description . . . . .	23
3.13.2 Member Data Documentation . . . . .	23
3.13.2.1 header . . . . .	23
3.13.2.2 vco_trim . . . . .	23
3.14 ModuleCfg Struct Reference . . . . .	23
3.14.1 Detailed Description . . . . .	23
3.15 ROUTING_TABLE_ITEM Struct Reference . . . . .	24
3.15.1 Detailed Description . . . . .	24
3.15.2 Member Data Documentation . . . . .	24
3.15.2.1 addr . . . . .	24
3.15.2.2 interface . . . . .	24
3.15.2.3 timer . . . . .	24
3.16 rssi_table_t Struct Reference . . . . .	25
3.16.1 Detailed Description . . . . .	25
3.16.2 Member Data Documentation . . . . .	25
3.16.2.1 addr . . . . .	25
3.16.2.2 rssi . . . . .	25
3.16.2.3 timer . . . . .	25
3.17 Timer_struct Struct Reference . . . . .	26
3.17.1 Detailed Description . . . . .	26
3.17.2 Member Data Documentation . . . . .	26
3.17.2.1 bootloader . . . . .	26
3.17.2.2 button . . . . .	26
3.17.2.3 buzz . . . . .	26
3.17.2.4 ms100 . . . . .	27
3.17.2.5 press_time . . . . .	27
3.17.2.6 sec . . . . .	27
3.17.2.7 timer32 . . . . .	27
3.17.2.8 timesync_broadcast . . . . .	27
3.18 TOA_DATA Struct Reference . . . . .	27
3.18.1 Detailed Description . . . . .	28

<b>4 File Documentation</b>	<b>29</b>
4.1 lib/common/ARM_Ethernet_Reader.h File Reference	29
4.1.1 Detailed Description	30
4.1.2 Enumeration Type Documentation	30
4.1.2.1 e_cfg_type	30
4.1.2.2 e_st_type	31
4.2 src/cfg.c File Reference	31
4.2.1 Detailed Description	32
4.2.2 Function Documentation	32
4.2.2.1 change_channel()	32
4.2.2.2 device_load_cfg_defaults()	32
4.2.2.3 device_load_configuration()	33
4.2.2.4 dw_load_cfg_defaults()	33
4.2.2.5 dw_load_configuration()	33
4.2.2.6 fix_version()	34
4.2.2.7 load_configuration()	34
4.2.2.8 ntr_load_cfg_defaults()	34
4.2.2.9 ntr_load_configuration()	35
4.2.2.10 save_configuration()	35
4.2.2.11 sx_load_cfg_defaults()	35
4.2.2.12 sx_load_configuration()	36
4.2.2.13 update_configuration()	36
4.3 src/cfg.h File Reference	36
4.3.1 Detailed Description	37
4.3.2 Function Documentation	37
4.3.2.1 fix_version()	37
4.3.2.2 load_configuration()	37
4.3.2.3 save_configuration()	39
4.3.2.4 update_configuration()	39
4.4 src/comm_test.c File Reference	39



4.4.1	Detailed Description	40
4.4.2	Function Documentation	40
4.4.2.1	ctest_answer_detected()	40
4.4.2.2	ctest_answer_tout()	41
4.4.2.3	ctest_data_generate()	41
4.4.2.4	ctest_get_result()	41
4.4.2.5	ctest_handle()	42
4.4.2.6	ctest_send()	42
4.4.2.7	ctest_start()	42
4.4.2.8	ctest_stop()	43
4.5	src/comm_test.h File Reference	43
4.5.1	Detailed Description	44
4.5.2	Function Documentation	44
4.5.2.1	ctest_answer_detected()	44
4.5.2.2	ctest_answer_tout()	44
4.5.2.3	ctest_get_result()	44
4.5.2.4	ctest_handle()	45
4.5.2.5	ctest_start()	45
4.5.2.6	ctest_stop()	45
4.6	src/config.h File Reference	46
4.6.1	Detailed Description	49
4.6.2	Macro Definition Documentation	49
4.6.2.1	USE_USB_PLL	49
4.7	src/extcomm.c File Reference	49
4.7.1	Detailed Description	51
4.7.2	Function Documentation	51
4.7.2.1	answer_handle()	51
4.7.2.2	answer_ms_tick()	51
4.7.2.3	answer_timeout_handle()	52
4.7.2.4	beacon_broadcast()	52

4.7.2.5	<a href="#">clear_answer_timer()</a>	52
4.7.2.6	<a href="#">dev_broadcast_handle()</a>	52
4.7.2.7	<a href="#">dev_handle()</a>	53
4.7.2.8	<a href="#">is_dev_handle()</a>	53
4.7.2.9	<a href="#">module1_handle()</a>	54
4.7.2.10	<a href="#">module2_handle()</a>	54
4.7.2.11	<a href="#">packet_route()</a>	54
4.7.2.12	<a href="#">send_wakeup_burst()</a>	55
4.7.2.13	<a href="#">set_answer_timer()</a>	55
4.7.2.14	<a href="#">sx1_pck_handle()</a>	56
4.7.2.15	<a href="#">sx2_pck_handle()</a>	56
4.7.2.16	<a href="#">tcp_pck_handle()</a>	56
4.7.2.17	<a href="#">uart_sys_pck_handle()</a>	57
4.7.2.18	<a href="#">udp_pck_handle()</a>	57
4.7.2.19	<a href="#">usb_pck_handle()</a>	57
4.8	<a href="#">src/extcomm.h File Reference</a>	57
4.8.1	<a href="#">Detailed Description</a>	58
4.8.2	<a href="#">Function Documentation</a>	58
4.8.2.1	<a href="#">answer_ms_tick()</a>	59
4.8.2.2	<a href="#">answer_timeout_handle()</a>	59
4.8.2.3	<a href="#">beacon_broadcast()</a>	59
4.8.2.4	<a href="#">clear_answer_timer()</a>	59
4.8.2.5	<a href="#">module1_handle()</a>	60
4.8.2.6	<a href="#">module2_handle()</a>	60
4.8.2.7	<a href="#">send_wakeup_burst()</a>	60
4.8.2.8	<a href="#">set_answer_timer()</a>	61
4.8.2.9	<a href="#">tcp_pck_handle()</a>	61
4.8.2.10	<a href="#">uart_sys_pck_handle()</a>	61
4.8.2.11	<a href="#">udp_pck_handle()</a>	62
4.8.2.12	<a href="#">usb_pck_handle()</a>	62

4.9	src/hardfault.c File Reference	62
4.9.1	Detailed Description	62
4.10	src/main.c File Reference	62
4.10.1	Detailed Description	64
4.10.2	Function Documentation	64
4.10.2.1	ADC_init()	64
4.10.2.2	Chip_USB_Init()	64
4.10.2.3	GPIO_IRQHandler()	65
4.10.2.4	main()	65
4.10.2.5	read_system_timer()	65
4.10.2.6	reinit_SX_modules()	65
4.10.2.7	SystemClkOut_init()	65
4.10.2.8	SystemCoreClock_init()	66
4.10.2.9	SysTick_Handler()	66
4.11	src/main.h File Reference	66
4.11.1	Detailed Description	67
4.11.2	Macro Definition Documentation	68
4.11.2.1	handle_led	68
4.11.2.2	handle_led_blink	68
4.11.2.3	handle_timer	70
4.11.2.4	handle_timer_flags	70
4.11.3	Function Documentation	71
4.11.3.1	Chip_USB_Init()	71
4.11.3.2	read_system_timer()	71
4.11.3.3	SystemClkOut_init()	71
4.12	src/main_bootloader.c File Reference	72
4.12.1	Detailed Description	72
4.12.2	Function Documentation	72
4.12.2.1	main()	72
4.13	src/modules.c File Reference	73

4.13.1 Detailed Description . . . . .	74
4.13.2 Macro Definition Documentation . . . . .	74
4.13.2.1 dw_select . . . . .	74
4.13.2.2 dw_unselect . . . . .	75
4.13.2.3 module_reset . . . . .	75
4.13.2.4 ntr_select . . . . .	75
4.13.2.5 ntr_unselect . . . . .	76
4.13.2.6 sx_select . . . . .	76
4.13.2.7 sx_unselect . . . . .	76
4.13.3 Function Documentation . . . . .	77
4.13.3.1 module_configure_pins() . . . . .	77
4.13.3.2 module_detect() . . . . .	77
4.13.3.3 module_dw_present() . . . . .	78
4.13.3.4 module_get_rssi() . . . . .	78
4.13.3.5 module_get_rssi_background() . . . . .	78
4.13.3.6 module_ntr_present() . . . . .	79
4.13.3.7 module_pck_send() . . . . .	79
4.13.3.8 module_pck_send_receive() . . . . .	79
4.13.3.9 module_set_pwr() . . . . .	80
4.13.3.10 module_set_sleep() . . . . .	80
4.13.3.11 module_sx_present() . . . . .	81
4.13.3.12 modules_init() . . . . .	81
4.14 src/modules.h File Reference . . . . .	81
4.14.1 Detailed Description . . . . .	82
4.14.2 Enumeration Type Documentation . . . . .	82
4.14.2.1 e_module . . . . .	82
4.14.2.2 e_module_type . . . . .	83
4.14.3 Function Documentation . . . . .	83
4.14.3.1 module_get_rssi() . . . . .	83
4.14.3.2 module_get_rssi_background() . . . . .	84

4.14.3.3	<code>module_pck_send()</code>	84
4.14.3.4	<code>module_pck_send_receive()</code>	84
4.14.3.5	<code>modules_init()</code>	85
4.15	<code>src/nanotron.c</code> File Reference	85
4.15.1	Detailed Description	86
4.15.2	Function Documentation	87
4.15.2.1	<code>ErrorHandler()</code>	87
4.15.2.2	<code>nanotron_beacon()</code>	87
4.15.2.3	<code>nanotron_broadcast_rtc()</code>	87
4.15.2.4	<code>nanotron_calibrate()</code>	88
4.15.2.5	<code>nanotron_calibrate_all()</code>	88
4.15.2.6	<code>nanotron_enable_promiscuous()</code>	88
4.15.2.7	<code>nanotron_get_ptr()</code>	89
4.15.2.8	<code>nanotron_get_toa()</code>	89
4.15.2.9	<code>nanotron_init()</code>	89
4.15.2.10	<code>nanotron_init_all()</code>	90
4.15.2.11	<code>nanotron_pwr_down()</code>	90
4.15.2.12	<code>nanotron_rx_enable()</code>	90
4.15.2.13	<code>nanotron_set_tx_pwr()</code>	91
4.15.2.14	<code>nanotron_task()</code>	91
4.15.2.15	<code>send_timesync_packet()</code>	92
4.16	<code>src/nanotron.h</code> File Reference	92
4.16.1	Detailed Description	93
4.16.2	Macro Definition Documentation	93
4.16.2.1	<code>SendMsg</code>	93
4.16.3	Function Documentation	93
4.16.3.1	<code>nanotron_beacon()</code>	94
4.16.3.2	<code>nanotron_broadcast_rtc()</code>	94
4.16.3.3	<code>nanotron_calibrate()</code>	94
4.16.3.4	<code>nanotron_calibrate_all()</code>	95

4.16.3.5	<a href="#">nanotron_enable_promiscuous()</a>	95
4.16.3.6	<a href="#">nanotron_get_ptr()</a>	95
4.16.3.7	<a href="#">nanotron_get_toa()</a>	96
4.16.3.8	<a href="#">nanotron_init()</a>	96
4.16.3.9	<a href="#">nanotron_init_all()</a>	96
4.16.3.10	<a href="#">nanotron_rx_enable()</a>	96
4.16.3.11	<a href="#">nanotron_set_tx_pwr()</a>	97
4.16.3.12	<a href="#">nanotron_task()</a>	97
4.17	<a href="#">src/pindef.c File Reference</a>	97
4.17.1	<a href="#">Detailed Description</a>	98
4.17.2	<a href="#">Function Documentation</a>	98
4.17.2.1	<a href="#">pindef_init()</a>	98
4.18	<a href="#">src/pindef.h File Reference</a>	98
4.18.1	<a href="#">Detailed Description</a>	102
4.18.2	<a href="#">Function Documentation</a>	102
4.18.2.1	<a href="#">pindef_init()</a>	102
4.19	<a href="#">src/power_ctrl.c File Reference</a>	102
4.19.1	<a href="#">Detailed Description</a>	103
4.19.2	<a href="#">Function Documentation</a>	103
4.19.2.1	<a href="#">avg_handle()</a>	103
4.19.2.2	<a href="#">pwr_handle()</a>	103
4.19.2.3	<a href="#">pwr_indication()</a>	104
4.19.2.4	<a href="#">pwr_meas_vals()</a>	104
4.19.2.5	<a href="#">pwr_suicide()</a>	104
4.19.2.6	<a href="#">set_chrg_mode()</a>	104
4.20	<a href="#">src/power_ctrl.h File Reference</a>	105
4.20.1	<a href="#">Detailed Description</a>	105
4.20.2	<a href="#">Enumeration Type Documentation</a>	105
4.20.2.1	<a href="#">e_chrg_mode</a>	105
4.20.3	<a href="#">Function Documentation</a>	106

4.20.3.1	<a href="#">pwr_handle()</a>	106
4.20.3.2	<a href="#">pwr_meas_vals()</a>	106
4.20.3.3	<a href="#">pwr_suicide()</a>	106
4.20.3.4	<a href="#">set_chrg_mode()</a>	107
4.21	<a href="#">src/routing.c File Reference</a>	107
4.21.1	<a href="#">Detailed Description</a>	107
4.21.2	<a href="#">Function Documentation</a>	108
4.21.2.1	<a href="#">routing_table_device_del()</a>	108
4.21.2.2	<a href="#">routing_table_find()</a>	109
4.21.2.3	<a href="#">routing_table_timeout()</a>	109
4.21.2.4	<a href="#">routing_table_update()</a>	109
4.22	<a href="#">src/routing.h File Reference</a>	110
4.22.1	<a href="#">Detailed Description</a>	111
4.22.2	<a href="#">Function Documentation</a>	111
4.22.2.1	<a href="#">routing_table_device_del()</a>	111
4.22.2.2	<a href="#">routing_table_find()</a>	111
4.22.2.3	<a href="#">routing_table_timeout()</a>	112
4.22.2.4	<a href="#">routing_table_update()</a>	112
4.23	<a href="#">src/rssi.c File Reference</a>	112
4.23.1	<a href="#">Detailed Description</a>	113
4.23.2	<a href="#">Function Documentation</a>	113
4.23.2.1	<a href="#">rssi_table_add()</a>	113
4.23.2.2	<a href="#">rssi_table_get()</a>	113
4.23.2.3	<a href="#">rssi_table_tick()</a>	114
4.24	<a href="#">src/rssi.h File Reference</a>	114
4.24.1	<a href="#">Detailed Description</a>	115
4.24.2	<a href="#">Function Documentation</a>	115
4.24.2.1	<a href="#">rssi_table_add()</a>	115
4.24.2.2	<a href="#">rssi_table_get()</a>	116
4.24.2.3	<a href="#">rssi_table_tick()</a>	116

4.25	src/spi_wrapper.c File Reference	116
4.25.1	Detailed Description	117
4.25.2	Function Documentation	117
4.25.2.1	SPI_Master_Init()	117
4.25.2.2	SPI_Read_Write()	117
4.25.2.3	SPI_Wait_Busy()	118
4.25.2.4	SPI_Write()	118
4.26	src/spi_wrapper.h File Reference	118
4.26.1	Detailed Description	119
4.26.2	Function Documentation	119
4.26.2.1	SPI_Master_Init()	119
4.26.2.2	SPI_Read_Write()	120
4.26.2.3	SPI_Wait_Busy()	120
4.26.2.4	SPI_Write()	121
<b>Index</b>		<b>123</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">AvgStruct</a>	
Averaging structure . . . . .	5
<a href="#">CommStruct</a>	
Parameters for answer detection . . . . .	6
<a href="#">CommTestResultStruct</a>	
Result of communication test . . . . .	7
<a href="#">CommTestSettStruct</a>	
Settings of communication test . . . . .	9
<a href="#">EthernetReader_Cfg</a>	
Device configuration structure . . . . .	10
<a href="#">EthernetReader_DW_Cfg</a>	
Decawave configuration structure . . . . .	11
<a href="#">EthernetReader_NTR_Cfg</a>	
Nanotron configuration structure . . . . .	12
<a href="#">EthernetReader_State</a>	
Device state structure . . . . .	13
<a href="#">EthernetReader_SX_Cfg</a>	
Semtech configuration structure . . . . .	16
<a href="#">Flag_struct</a>	
Flags . . . . .	18
<a href="#">Leds_struct</a>	
Variables for LED lighting and blinking . . . . .	20
<a href="#">MODULE_STRUCT</a>	
Module state in each slot . . . . .	22
<a href="#">Module_SX_Cfg</a>	
Configuration in SX module EEPROM . . . . .	22
<a href="#">ModuleCfg</a>	
One slot configuration . . . . .	23
<a href="#">ROUTING_TABLE_ITEM</a>	
Routing table item structure . . . . .	24
<a href="#">rssi_table_t</a>	
RSSI table item structure . . . . .	25
<a href="#">Timer_struct</a>	
Timers . . . . .	26
<a href="#">TOA_DATA</a>	
ToA (Time of Arrival) structure . . . . .	27



## Chapter 2

# File Index

### 2.1 File List

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

lib/common/ <a href="#">ARM_Ethernet_Reader.h</a>	
Device configuration structures and requests definitions	29
src/ <a href="#">cfg.c</a>	31
src/ <a href="#">cfg.h</a>	
Device configuration functions for its loading, saving and updating	36
src/ <a href="#">comm_test.c</a>	39
src/ <a href="#">comm_test.h</a>	
Testing communication between two devices via wireless network	43
src/ <a href="#">config.h</a>	
Configuration and macro definitions for libraries	46
src/ <a href="#">extcomm.c</a>	49
src/ <a href="#">extcomm.h</a>	
Extended communication functions	57
src/ <a href="#">hardfault.c</a>	62
src/ <a href="#">main.c</a>	62
src/ <a href="#">main.h</a>	
Function and structures defined in main program	66
src/ <a href="#">main_bootloader.c</a>	72
src/ <a href="#">modules.c</a>	73
src/ <a href="#">modules.h</a>	
Modules control	81
src/ <a href="#">nanotron.c</a>	85
src/ <a href="#">nanotron.h</a>	
Functions for controlling NanoLOC	92
src/ <a href="#">pindef.c</a>	97
src/ <a href="#">pindef.h</a>	
Pin definitions and configuration	98
src/ <a href="#">power_ctrl.c</a>	102
src/ <a href="#">power_ctrl.h</a>	
Power and charging control	105
src/ <a href="#">routing.c</a>	107
src/ <a href="#">routing.h</a>	
Routing function	110
src/ <a href="#">rssi.c</a>	112
src/ <a href="#">rssi.h</a>	
Receive Signal Strength Intensity Table	114

src/ <a href="#">spi_wrapper.c</a> . . . . .	116
src/ <a href="#">spi_wrapper.h</a> Wrapper for using more SPIs . . . . .	118

## Chapter 3

# Class Documentation

### 3.1 AvgStruct Struct Reference

Averaging structure.

#### Public Attributes

- unsigned long [sum](#)
- unsigned char [samples](#)

#### 3.1.1 Detailed Description

Averaging structure.

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 samples

```
unsigned char AvgStruct::samples
```

Number of sum samples

##### 3.1.2.2 sum

```
unsigned long AvgStruct::sum
```

Sum of input values

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

- [src/power\\_ctrl.c](#)

## 3.2 CommStruct Struct Reference

Parameters for answer detection.

```
#include <extcomm.h>
```

### Public Attributes

- volatile unsigned int [addr\\_dest](#)
- volatile unsigned char [source\\_dev](#)
- volatile unsigned char [answer\\_id](#)
- volatile unsigned char [sent\\_request](#)
- volatile unsigned char [send\\_tries](#)
- volatile unsigned int [answer\\_tmr](#)
- volatile bool [awaiting\\_answer](#)
- volatile bool [answer\\_timeout\\_flag](#)

### 3.2.1 Detailed Description

Parameters for answer detection.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 [addr\\_dest](#)

```
volatile unsigned int CommStruct::addr_dest
```

Destination device address

#### 3.2.2.2 [answer\\_id](#)

```
volatile unsigned char CommStruct::answer_id
```

Packet ID

#### 3.2.2.3 [answer\\_timeout\\_flag](#)

```
volatile bool CommStruct::answer_timeout_flag
```

Answer timeout flag

#### 3.2.2.4 answer\_tmr

```
volatile unsigned int CommStruct::answer_tmr
```

Awaiting answer timer

#### 3.2.2.5 awaiting\_answer

```
volatile bool CommStruct::awaiting_answer
```

Awaiting answer flag

#### 3.2.2.6 send\_tries

```
volatile unsigned char CommStruct::send_tries
```

Number of tries send packet

#### 3.2.2.7 sent\_request

```
volatile unsigned char CommStruct::sent_request
```

Packet request

#### 3.2.2.8 source\_dev

```
volatile unsigned char CommStruct::source_dev
```

Source device address

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

- [src/extcomm.h](#)

## 3.3 CommTestResultStruct Struct Reference

Result of communication test.

```
#include <comm_test.h>
```

### Public Attributes

- unsigned [in\\_progress](#): 1
- unsigned [reserved](#): 7
- volatile unsigned char [completed](#)
- volatile unsigned short [ok](#)
- volatile unsigned short [nok](#)

### 3.3.1 Detailed Description

Result of communication test.

### 3.3.2 Member Data Documentation

#### 3.3.2.1 completed

```
volatile unsigned char CommTestResultStruct::completed
```

Communication test progress in [%]

#### 3.3.2.2 in\_progress

```
unsigned CommTestResultStruct::in_progress
```

Indicating active test

#### 3.3.2.3 nok

```
volatile unsigned short CommTestResultStruct::nok
```

Number of bad packets

#### 3.3.2.4 ok

```
volatile unsigned short CommTestResultStruct::ok
```

Number of correctly sendet packets

#### 3.3.2.5 reserved

```
unsigned CommTestResultStruct::reserved
```

Bits reserved for future use

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

- [src/comm\\_test.h](#)



## 3.4 CommTestSettStruct Struct Reference

Settings of communication test.

```
#include <comm_test.h>
```

### Public Attributes

- volatile unsigned int [addr](#)
- volatile unsigned char [data\\_len](#)
- volatile unsigned short [cnt](#)
- volatile unsigned short [timeout](#)

### 3.4.1 Detailed Description

Settings of communication test.

### 3.4.2 Member Data Documentation

#### 3.4.2.1 addr

```
volatile unsigned int CommTestSettStruct::addr
```

Address of device for communication test

#### 3.4.2.2 cnt

```
volatile unsigned short CommTestSettStruct::cnt
```

Number of packets which will be send

#### 3.4.2.3 data\_len

```
volatile unsigned char CommTestSettStruct::data_len
```

Length of data in one packet

#### 3.4.2.4 timeout

```
volatile unsigned short CommTestSettStruct::timeout
```

Timeout for one send packet in [ms]

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

- [src/comm\\_test.h](#)

## 3.5 EthernetReader\_Cfg Struct Reference

Device configuration structure.

```
#include <ARM_Ethernet_Reader.h>
```

### Public Attributes

- `STRUCT_HEADER` [header](#)
- unsigned [module1\\_on](#): 1
- unsigned [module2\\_on](#): 1
- unsigned [route\\_from\\_wireless1](#): 1
- unsigned [route\\_from\\_wireless2](#): 1
- unsigned [usb\\_enabled](#): 1
- unsigned [nanotron\\_clk\\_en](#): 1
- unsigned [reserved](#): 2

### 3.5.1 Detailed Description

Device configuration structure.

### 3.5.2 Member Data Documentation

#### 3.5.2.1 header

```
STRUCT_HEADER EthernetReader_Cfg::header
```

Header of structure

#### 3.5.2.2 module1\_on

```
unsigned EthernetReader_Cfg::module1_on
```

Enable power to module 1

#### 3.5.2.3 module2\_on

```
unsigned EthernetReader_Cfg::module2_on
```

Enable power to module 2

#### 3.5.2.4 nanotron\_clk\_en

```
unsigned EthernetReader_Cfg::nanotron_clk_en
```

Enable 32 MHz clock for nanotron

#### 3.5.2.5 reserved

```
unsigned EthernetReader_Cfg::reserved
```

Reserved for future use

#### 3.5.2.6 route\_from\_wireless1

```
unsigned EthernetReader_Cfg::route_from_wireless1
```

Routing all packet from module 1

#### 3.5.2.7 route\_from\_wireless2

```
unsigned EthernetReader_Cfg::route_from_wireless2
```

Routing all packet from module 2

#### 3.5.2.8 usb\_enabled

```
unsigned EthernetReader_Cfg::usb_enabled
```

Enable USB communication.

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

- lib/common/[ARM\\_Ethernet\\_Reader.h](#)

## 3.6 EthernetReader\_DW\_Cfg Struct Reference

Decawave configuration structure.

```
#include <ARM_Ethernet_Reader.h>
```

### Public Attributes

- STRUCT\_HEADER [header](#)

### 3.6.1 Detailed Description

Decawave configuration structure.

#### Note

Not fully implemented yet

### 3.6.2 Member Data Documentation

#### 3.6.2.1 header

```
STRUCT_HEADER EthernetReader_DW_Cfg::header
```

Header of structure

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

- lib/common/[ARM\\_Ethernet\\_Reader.h](#)

## 3.7 EthernetReader\_NTR\_Cfg Struct Reference

Nanotron configuration structure.

```
#include <ARM_Ethernet_Reader.h>
```

### Public Attributes

- STRUCT\_HEADER [header](#)
- unsigned char [nano\\_pwr](#)
- unsigned char [mode](#)

### 3.7.1 Detailed Description

Nanotron configuration structure.

### 3.7.2 Member Data Documentation

### 3.7.2.1 header

```
STRUCT_HEADER EthernetReader_NTR_Cfg::header
```

Header of structure

### 3.7.2.2 mode

```
unsigned char EthernetReader_NTR_Cfg::mode
```

Mode of Nanotron configuration

### 3.7.2.3 nano\_pwr

```
unsigned char EthernetReader_NTR_Cfg::nano_pwr
```

Transmitting power

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

- lib/common/[ARM\\_Ethernet\\_Reader.h](#)

## 3.8 EthernetReader\_State Struct Reference

Device state structure.

```
#include <ARM_Ethernet_Reader.h>
```

### Public Attributes

- STRUCT\_HEADER [header](#)
- uint32\_t [uptime](#)
- uint16\_t [supply\\_voltage](#)
- uint16\_t [usb\\_voltage](#)
- uint16\_t [batt\\_voltage](#)
- uint16\_t [chrg\\_current](#)
- TEMP [temperature](#)
- unsigned [charging](#): 1
- unsigned [pwr\\_good](#): 1
- unsigned [chrg\\_mode](#): 3
- unsigned [server\\_connected](#): 1
- unsigned [reserved](#): 2
- uint8\_t [module1\\_type](#)
- uint8\_t [module2\\_type](#)
- uint8\_t [module1\\_rssi\\_back](#)
- uint8\_t [module2\\_rssi\\_back](#)

### 3.8.1 Detailed Description

Device state structure.

### 3.8.2 Member Data Documentation

#### 3.8.2.1 batt\_voltage

```
uint16_t EthernetReader_State::batt_voltage
```

Battery voltage [mV]

#### 3.8.2.2 charging

```
unsigned EthernetReader_State::charging
```

Charging active flag

#### 3.8.2.3 chrg\_current

```
uint16_t EthernetReader_State::chrg_current
```

Charging current [mA]

#### 3.8.2.4 chrg\_mode

```
unsigned EthernetReader_State::chrg_mode
```

Type of charging mode

#### 3.8.2.5 header

```
STRUCT_HEADER EthernetReader_State::header
```

Header of structure

#### 3.8.2.6 module1\_rssi\_back

```
uint8_t EthernetReader_State::module1_rssi_back
```

RSSI background from module 1

#### 3.8.2.7 module1\_type

```
uint8_t EthernetReader_State::module1_type
```

Type of module in slot 1

#### 3.8.2.8 module2\_rssi\_back

```
uint8_t EthernetReader_State::module2_rssi_back
```

RSSI background from module 2

#### 3.8.2.9 module2\_type

```
uint8_t EthernetReader_State::module2_type
```

Type of module in slot 2

#### 3.8.2.10 pwr\_good

```
unsigned EthernetReader_State::pwr_good
```

External power flag

#### 3.8.2.11 reserved

```
unsigned EthernetReader_State::reserved
```

Reserved for future use

#### 3.8.2.12 server\_connected

```
unsigned EthernetReader_State::server_connected
```

Server connection flag

#### 3.8.2.13 supply\_voltage

```
uint16_t EthernetReader_State::supply_voltage
```

Supply voltage (Before switching DC-DC converter) [mV]

#### 3.8.2.14 temperature

```
TEMP EthernetReader_State::temperature
```

Temperature [0.01 °C]

### 3.8.2.15 uptime

```
uint32_t EthernetReader_State::uptime
```

Device up-time [sec]

### 3.8.2.16 usb\_voltage

```
uint16_t EthernetReader_State::usb_voltage
```

Voltage on USB port [mV]

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

- lib/common/[ARM\\_Ethernet\\_Reader.h](#)

## 3.9 EthernetReader\_SX\_Cfg Struct Reference

Semtech configuration structure.

```
#include <ARM_Ethernet_Reader.h>
```

### Public Attributes

- STRUCT\_HEADER [header](#)
- unsigned long [rf\\_frequency\\_1](#)
- unsigned long [rf\\_frequency\\_2](#)
- unsigned char [if\\_gain](#)
- unsigned char [rf\\_power](#)
- unsigned char [vco\\_trim](#)
- unsigned [listen\\_on\\_chn2](#): 1
- unsigned [clk\\_out](#): 1
- unsigned [reserved](#): 6

### 3.9.1 Detailed Description

Semtech configuration structure.

### 3.9.2 Member Data Documentation

#### 3.9.2.1 clk\_out

```
unsigned EthernetReader_SX_Cfg::clk_out
```

Enable output clock (for measuring)



### 3.9.2.2 header

```
STRUCT_HEADER EthernetReader_SX_Cfg::header
```

Header of structure

### 3.9.2.3 if\_gain

```
unsigned char EthernetReader_SX_Cfg::if_gain
```

Gain of receiver amplifier

### 3.9.2.4 listen\_on\_chn2

```
unsigned EthernetReader_SX_Cfg::listen_on_chn2
```

Configuration selection

### 3.9.2.5 reserved

```
unsigned EthernetReader_SX_Cfg::reserved
```

Reserved for future use

### 3.9.2.6 rf\_frequency\_1

```
unsigned long EthernetReader_SX_Cfg::rf_frequency_1
```

Frequency of first configuration

### 3.9.2.7 rf\_frequency\_2

```
unsigned long EthernetReader_SX_Cfg::rf_frequency_2
```

Frequency of second configuration

### 3.9.2.8 rf\_power

```
unsigned char EthernetReader_SX_Cfg::rf_power
```

Transmitting power

### 3.9.2.9 vco\_trim

```
unsigned char EthernetReader_SX_Cfg::vco_trim
```

VCO configuration

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

- lib/common/[ARM\\_Ethernet\\_Reader.h](#)

## 3.10 Flag\_struct Struct Reference

Flags.

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

### Public Attributes

- volatile bool [sec](#)
- volatile bool [ms100](#)
- volatile bool [power\\_off](#)
- volatile bool [update\\_cfg](#)
- volatile bool [wakeup\\_burst](#)
- volatile bool [reinit\\_modules](#)
- volatile bool [usb\\_forced](#)
- volatile bool [sx1\\_irq\\_detected](#)
- volatile bool [sx2\\_irq\\_detected](#)

### 3.10.1 Detailed Description

Flags.

### 3.10.2 Member Data Documentation

#### 3.10.2.1 ms100

```
volatile bool Flag_struct::ms100
```

100 milliseconds flag. Set to 1 after each 100 milliseconds

#### 3.10.2.2 power\_off

```
volatile bool Flag_struct::power_off
```

Power off flag. Set to 1 shutdown device

### 3.10.2.3 reinit\_modules

```
volatile bool Flag_struct::reinit_modules
```

Reinitialization modules flag. Set to 1 reinit. all modules

### 3.10.2.4 sec

```
volatile bool Flag_struct::sec
```

Second flag. Set to 1 after each second

### 3.10.2.5 sx1\_irq\_detected

```
volatile bool Flag_struct::sx1_irq_detected
```

Semtech 1 interrupt flag. Set to 1 when interrupt detected

### 3.10.2.6 sx2\_irq\_detected

```
volatile bool Flag_struct::sx2_irq_detected
```

Semtech 2 interrupt flag. Set to 1 when interrupt detected

### 3.10.2.7 update\_cfg

```
volatile bool Flag_struct::update_cfg
```

Configuration update flag. Set to 1 updates all configuration

### 3.10.2.8 usb\_forced

```
volatile bool Flag_struct::usb_forced
```

USB forced flag. When is set, USB is connected and Ethernet disabled

### 3.10.2.9 wakeup\_burst

```
volatile bool Flag_struct::wakeup_burst
```

Wakeup burst flag. When set, burst sending is active

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

- [src/main.h](#)

## 3.11 Leds\_struct Struct Reference

Variables for LED lighting and blinking.

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

### Public Attributes

- unsigned [mod1\\_blink](#): 1
- unsigned [mod2\\_blink](#): 1
- unsigned [conn\\_blink](#): 1
- unsigned [reserved](#): 5
- volatile unsigned int [mod1](#)
- volatile unsigned int [mod2](#)
- volatile unsigned int [conn](#)
- volatile unsigned int [err](#)
- volatile unsigned int [pwr](#)
- volatile unsigned int [sys](#)

### 3.11.1 Detailed Description

Variables for LED lighting and blinking.

### 3.11.2 Member Data Documentation

#### 3.11.2.1 conn

```
volatile unsigned int Leds_struct::conn
```

Timer for Connection LED lighting

#### 3.11.2.2 conn\_blink

```
unsigned Leds_struct::conn_blink
```

Connection LED blink. Set to 1 perform one LED blink

#### 3.11.2.3 err

```
volatile unsigned int Leds_struct::err
```

Timer for Error LED lighting

#### 3.11.2.4 mod1

```
volatile unsigned int Leds_struct::mod1
```

Timer for Module 1 LED lighting

#### 3.11.2.5 mod1\_blink

```
unsigned Leds_struct::mod1_blink
```

Module 1 LED blink. Set to 1 perform one LED blink

#### 3.11.2.6 mod2

```
volatile unsigned int Leds_struct::mod2
```

Timer for Module 2 LED lighting

#### 3.11.2.7 mod2\_blink

```
unsigned Leds_struct::mod2_blink
```

Module 2 LED blink. Set to 1 perform one LED blink

#### 3.11.2.8 pwr

```
volatile unsigned int Leds_struct::pwr
```

Timer for Power LED lighting

#### 3.11.2.9 reserved

```
unsigned Leds_struct::reserved
```

Reserved for future use

#### 3.11.2.10 sys

```
volatile unsigned int Leds_struct::sys
```

Timer for System LED lighting

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

- [src/main.h](#)

## 3.12 MODULE\_STRUCT Struct Reference

Module state in each slot.

```
#include <modules.h>
```

### Public Attributes

- enum [e\\_module\\_type](#) slot1\_type
- enum [e\\_module\\_type](#) slot2\_type

### 3.12.1 Detailed Description

Module state in each slot.

### 3.12.2 Member Data Documentation

#### 3.12.2.1 slot1\_type

```
enum e\_module\_type MODULE_STRUCT::slot1_type
```

Module state in slot 1

#### 3.12.2.2 slot2\_type

```
enum e\_module\_type MODULE_STRUCT::slot2_type
```

Module state in slot 2

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

- src/[modules.h](#)

## 3.13 Module\_SX\_Cfg Struct Reference

Configuration in SX module EEPROM.

```
#include <ARM_Ethernet_Reader.h>
```

### Public Attributes

- STRUCT\_HEADER [header](#)
- unsigned char [vco\\_trim](#)

### 3.13.1 Detailed Description

Configuration in SX module EEPROM.

### 3.13.2 Member Data Documentation

#### 3.13.2.1 header

```
STRUCT_HEADER Module_SX_Cfg::header
```

Header of structure

#### 3.13.2.2 vco\_trim

```
unsigned char Module_SX_Cfg::vco_trim
```

VCO configuration

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

- lib/common/[ARM\\_Ethernet\\_Reader.h](#)

## 3.14 ModuleCfg Struct Reference

One slot configuration.

```
#include <ARM_Ethernet_Reader.h>
```

### Public Attributes

- [EthernetReader\\_SX\\_Cfg](#) **sx**
- [EthernetReader\\_NTR\\_Cfg](#) **ntr**
- [EthernetReader\\_DW\\_Cfg](#) **dw**

### 3.14.1 Detailed Description

One slot configuration.

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

- lib/common/[ARM\\_Ethernet\\_Reader.h](#)

## 3.15 ROUTING\_TABLE\_ITEM Struct Reference

Routing table item structure.

```
#include <routing.h>
```

### Public Attributes

- unsigned int [addr](#)
- enum e\_packet\_source\_device [interface](#)
- unsigned char [timer](#)

### 3.15.1 Detailed Description

Routing table item structure.

### 3.15.2 Member Data Documentation

#### 3.15.2.1 [addr](#)

```
unsigned int ROUTING_TABLE_ITEM::addr
```

Device address

#### 3.15.2.2 [interface](#)

```
enum e_packet_source_device ROUTING_TABLE_ITEM::interface
```

Interface where is the device present

#### 3.15.2.3 [timer](#)

```
unsigned char ROUTING_TABLE_ITEM::timer
```

Timer for time out this item

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

- [src/routing.h](#)



## 3.16 rssi\_table\_t Struct Reference

RSSI table item structure.

```
#include <rssi.h>
```

### Public Attributes

- unsigned int [addr](#)
- unsigned char [rssi](#) [2]
- unsigned char [timer](#) [2]

### 3.16.1 Detailed Description

RSSI table item structure.

### 3.16.2 Member Data Documentation

#### 3.16.2.1 addr

```
unsigned int rssi_table_t::addr
```

Device address

#### 3.16.2.2 rssi

```
unsigned char rssi_table_t::rssi[2]
```

RSSI for each module

#### 3.16.2.3 timer

```
unsigned char rssi_table_t::timer[2]
```

timer for item timeout for each module

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

- [src/rssi.h](#)

## 3.17 Timer\_struct Struct Reference

Timers.

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

### Public Attributes

- volatile unsigned int [sec](#)
- volatile unsigned char [ms100](#)
- volatile unsigned int [timer32](#)
- volatile unsigned int [buzz](#)
- volatile unsigned int [button](#)
- volatile unsigned int [press\\_time](#)
- volatile unsigned char [bootloader](#)
- volatile unsigned int [timesync\\_broadcast](#)

### 3.17.1 Detailed Description

Timers.

### 3.17.2 Member Data Documentation

#### 3.17.2.1 bootloader

```
volatile unsigned char Timer_struct::bootloader
```

Timer for run bootloader

#### 3.17.2.2 button

```
volatile unsigned int Timer_struct::button
```

Timer for measuring button press time

#### 3.17.2.3 buzz

```
volatile unsigned int Timer_struct::buzz
```

Timer for beeping

#### 3.17.2.4 ms100

```
volatile unsigned char Timer_struct::ms100
```

100 millisecond timer. For handling 100ms flag

#### 3.17.2.5 press\_time

```
volatile unsigned int Timer_struct::press_time
```

Result of button press time

#### 3.17.2.6 sec

```
volatile unsigned int Timer_struct::sec
```

Second timer. For handling second flag

#### 3.17.2.7 timer32

```
volatile unsigned int Timer_struct::timer32
```

32 bit timer incremented each millisecond

#### 3.17.2.8 timesync\_broadcast

```
volatile unsigned int Timer_struct::timesync_broadcast
```

Timer for broadcasting time through nanotron

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

- [src/main.h](#)

## 3.18 TOA\_DATA Struct Reference

ToA (Time of Arrival) structure.

```
#include <nanotron.h>
```

### Public Attributes

- `uint16_t timeslot`
- `uint8_t ToaOffsetMeanDataValid`
- `uint8_t PhaseOffsetData`
- `uint16_t ToaOffsetMeanData`
- `uint64_t BaseTimer`
- `uint8_t AgcGain`
- `uint64_t RTCSecs`
- `int16_t RTCMsecs`

### 3.18.1 Detailed Description

ToA (Time of Arrival) structure.

#### Note

For ranging functionality implementation

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

- [src/nanotron.h](#)

## Chapter 4

# File Documentation

### 4.1 lib/common/ARM\_Ethernet\_Reader.h File Reference

Device configuration structures and requets definitions.

```
#include <type.h>
#include <coremem.h>
```

#### Classes

- struct [EthernetReader\\_State](#)  
*Device state structure.*
- struct [EthernetReader\\_Cfg](#)  
*Device configuration structure.*
- struct [EthernetReader\\_SX\\_Cfg](#)  
*Semtech configuration structure.*
- struct [EthernetReader\\_NTR\\_Cfg](#)  
*Nanotron configuration structure.*
- struct [EthernetReader\\_DW\\_Cfg](#)  
*Decawave configuration structure.*
- struct [Module\\_SX\\_Cfg](#)  
*Configuration in SX module EEPROM.*
- struct [ModuleCfg](#)  
*One slot configuration.*

#### Macros

- #define **RQ\_SERVICE\_SET\_FREQ** 0x01
- #define **RQ\_SERVICE\_SHUTDOWN** 0x02
- #define **RQ\_SERVICE\_CTEST\_START** 0x03
- #define **RQ\_SERVICE\_CTEST\_STOP** 0x04
- #define **RQ\_SERVICE\_CTEST\_GET\_RESULT** 0x05
- #define **RQ\_SERVICE\_GET\_RSSI\_TABLE** 0x06
- #define [ST\\_VER\\_DEVICE](#) 2

- *Device state structure version.*  
• #define `CFG_VER_DEVICE` 1  
*Version of device configuration structure.*
- #define `CFG_VER_SX` 1  
*Version of Semtech configuration structure.*
- #define `CFG_VER_NTR` 1  
*Version of Nanotron configuration structure.*
- #define `CFG_VER_DW` 1  
*Version of Decawave configuration structure.*
- #define `CFG_VER_SX_MODULE` 1  
*Version of configuration in SX module EEPROM.*

## Enumerations

- enum `e_st_type` { `ST_DEVICE` }  
*Selection type of status structure.*
- enum `e_cfg_type` {  
`CFG_DEVICE`, `CFG_MODULE1_SX`, `CFG_MODULE1_NTR`, `CFG_MODULE1_DW`,  
`CFG_MODULE2_SX`, `CFG_MODULE2_NTR`, `CFG_MODULE2_DW`, `CFG_ALL` = 0xFF }  
*Selection type of configuration structure.*

### 4.1.1 Detailed Description

Device configuration structures and requests definitions.

#### Author

Ondrej Jerabek

### 4.1.2 Enumeration Type Documentation

#### 4.1.2.1 e\_cfg\_type

enum `e_cfg_type`

Selection type of configuration structure.

#### Enumerator

<code>CFG_DEVICE</code>	Device configuration
<code>CFG_MODULE1_SX</code>	Semtech module 1 configuration
<code>CFG_MODULE1_NTR</code>	Nanotron module 1 configuration
<code>CFG_MODULE1_DW</code>	Decawave module 1 configuration
<code>CFG_MODULE2_SX</code>	Semtech module 2 configuration
<code>CFG_MODULE2_NTR</code>	Nanotron module 2 configuration
<code>CFG_MODULE2_DW</code>	Decawave module 2 configuration
<code>CFG_ALL</code>	All configuration selection

## 4.1.2.2 e\_st\_type

enum [e\\_st\\_type](#)

Selection type of status structure.

Enumerator

ST_DEVICE	Device status
-----------	---------------

## 4.2 src/cfg.c File Reference

```
#include "config.h"
#include "cfg.h"
#include "main.h"
#include "ARM_Ethernet_Reader.h"
#include "modules.h"
#include "nanotron.h"
#include <coremem.h>
#include <stddef.h>
#include <eeprom.h>
#include <eepromcrc.h>
#include <macros.h>
#include <stdio.h>
#include <sx1211config.h>
#include <multisx1211.h>
#include <multisx1211_packet.h>
#include <config-default.h>
```

### Functions

- void [fix\\_version](#) (STRUCT\_HEADER \*header, enum [e\\_cfg\\_type](#) cfg\_type, unsigned char cfg\_ver)  
*Fix version of configuration specified by parameter cfg\_type.*
- void [device\\_load\\_cfg\\_defaults](#) (void)  
*Load default device configuration.*
- void [sx\\_load\\_cfg\\_defaults](#) (EthernetReader\_SX\_Cfg \*p\_cfg)  
*Load default Semtech module configuration.*
- void [ntr\\_load\\_cfg\\_defaults](#) (EthernetReader\_NTR\_Cfg \*p\_cfg)  
*Load default Nanotron module configuration.*
- void [dw\\_load\\_cfg\\_defaults](#) (EthernetReader\_DW\_Cfg \*p\_cfg)  
*Load default Decawave module configuration.*
- void [device\\_load\\_configuration](#) (void)  
*Load device configuration from EEPROM memory and fill device info string.*
- void [sx\\_load\\_configuration](#) (EthernetReader\_SX\_Cfg \*p\_cfg, unsigned int addr\_eeprom)  
*Load Semtech module configuration from EEPROM memory.*
- void [ntr\\_load\\_configuration](#) (EthernetReader\_NTR\_Cfg \*p\_cfg, unsigned int addr\_eeprom)

- Load Nanotron module configuration from EEPROM memory.*
- void [dw\\_load\\_configuration](#) ([EthernetReader\\_DW\\_Cfg](#) \*p\_cfg, unsigned int addr\_eeprom)
- Load Decawave module configuration from EEPROM memory.*
- void [change\\_channel](#) (enum [e\\_module](#) module)
- Update active channel configuration on Semtech module.*
- void [load\\_configuration](#) (enum [e\\_cfg\\_type](#) cfg\_type)
- Load configuration specified by parameter cfg\_type.*
- void [save\\_configuration](#) (enum [e\\_cfg\\_type](#) cfg\_type)
- Save configuration specified by parameter cfg\_type.*
- void [update\\_configuration](#) ()
- Update all device configurations.*

## 4.2.1 Detailed Description

### Author

Ondrej Jerabek

## 4.2.2 Function Documentation

### 4.2.2.1 [change\\_channel\(\)](#)

```
void change_channel (
    enum e\_module module )
```

Update active channel configuration on Semtech module.

#### Parameters

<i>module</i>	Module index
---------------	--------------

#### Returns

Nothing

### 4.2.2.2 [device\\_load\\_cfg\\_defaults\(\)](#)

```
void device_load_cfg_defaults (
    void )
```

Load default device configuration.

#### Returns

Nothing



#### 4.2.2.3 device\_load\_configuration()

```
void device_load_configuration (
    void )
```

Load device configuration from EEPROM memory and fill device info string.

##### Returns

Nothing

#### 4.2.2.4 dw\_load\_cfg\_defaults()

```
void dw_load_cfg_defaults (
    EthernetReader_DW_Cfg * p_cfg )
```

Load default Decawave module configuration.

##### Parameters

out	<i>p_cfg</i>	Pointer to Decawave configuration
-----	--------------	-----------------------------------

##### Returns

Nothing

#### 4.2.2.5 dw\_load\_configuration()

```
void dw_load_configuration (
    EthernetReader_DW_Cfg * p_cfg,
    unsigned int addr_eeprom )
```

Load Decawave module configuration from EEPROM memory.

##### Parameters

out	<i>p_cfg</i>	Pointer to Decawave configuration
	<i>addr_eeprom</i>	EEPROM address where the configuration is located

##### Returns

Nothing

#### 4.2.2.6 fix\_version()

```
void fix_version (
    STRUCT_HEADER * header,
    enum e_cfg_type cfg_type,
    unsigned char cfg_ver )
```

Fix version of configuration specified by parameter `cfg_type`.

##### Parameters

<i>header</i>	Pointer to header of configuration
<i>cfg_type</i>	Type of configuration which will be fixed
<i>cfg_ver</i>	Version of configuration

##### Returns

Nothing

#### 4.2.2.7 load\_configuration()

```
void load_configuration (
    enum e_cfg_type cfg_type )
```

Load configuration specified by parameter `cfg_type`.

##### Parameters

<i>cfg_type</i>	Type of configuration which will be loaded
-----------------	--

##### Returns

Nothing

#### 4.2.2.8 ntr\_load\_cfg\_defaults()

```
void ntr_load_cfg_defaults (
    EthernetReader_NTR_Cfg * p_cfg )
```

Load default Nanotron module configuration.

##### Parameters

out	<i>p_cfg</i>	Pointer to Nanotron configuration
-----	--------------	-----------------------------------

**Returns**

Nothing

**4.2.2.9 ntr\_load\_configuration()**

```
void ntr_load_configuration (
    EthernetReader_NTR_Cfg * p_cfg,
    unsigned int addr_eeprom )
```

Load Nanotron module configuration from EEPROM memory.

**Parameters**

out	<i>p_cfg</i>	Pointer to Nanotron configuration
	<i>addr_eeprom</i>	EEPROM address where the configuration is located

**Returns**

Nothing

**4.2.2.10 save\_configuration()**

```
void save_configuration (
    enum e_cfg_type cfg_type )
```

Save configuration specified by parameter *cfg\_type*.

**Parameters**

<i>cfg_type</i>	Type of configuration which will be saved
-----------------	---

**Returns**

Nothing

**4.2.2.11 sx\_load\_cfg\_defaults()**

```
void sx_load_cfg_defaults (
    EthernetReader_SX_Cfg * p_cfg )
```

Load default Semtech module configuration.

**Parameters**

out	<i>p_cfg</i>	Pointer to Semtech configuration
-----	--------------	----------------------------------

**Returns**

Nothing

**4.2.2.12 sx\_load\_configuration()**

```
void sx_load_configuration (
    EthernetReader_SX_Cfg * p_cfg,
    unsigned int addr_eeprom )
```

Load Semtech module configuration from EEPROM memory.

**Parameters**

out	<i>p_cfg</i>	Pointer to Semtech configuration
	<i>addr_eeprom</i>	EEPROM address where the configuration is located

**Returns**

Nothing

**4.2.2.13 update\_configuration()**

```
void update_configuration (
    void )
```

Update all device configurations.

**Returns**

Nothing

**4.3 src/cfg.h File Reference**

Device configuration functions for its loading, saving and updating.

```
#include <type.h>
#include <rqstat.h>
#include <coremem.h>
#include "ARM_Ethernet_Reader.h"
#include "config.h"
```

## Functions

- void [save\\_configuration](#) (enum [e\\_cfg\\_type](#) cfg\_type)  
*Save configuration specified by parameter cfg\_type.*
- void [load\\_configuration](#) (enum [e\\_cfg\\_type](#) cfg\_type)  
*Load configuration specified by parameter cfg\_type.*
- void [update\\_configuration](#) (void)  
*Update all device configurations.*
- void [fix\\_version](#) (STRUCT\_HEADER \*header, enum [e\\_cfg\\_type](#) cfg\_type, unsigned char cfg\_ver)  
*Fix version of configuration specified by parameter cfg\_type.*

### 4.3.1 Detailed Description

Device configuration functions for its loading, saving and updating.

#### Author

Ondrej Jerabek

### 4.3.2 Function Documentation

#### 4.3.2.1 [fix\\_version\(\)](#)

```
void fix_version (
    STRUCT_HEADER * header,
    enum e\_cfg\_type cfg_type,
    unsigned char cfg_ver )
```

Fix version of configuration specified by parameter cfg\_type.

#### Parameters

<i>header</i>	Pointer to header of configuration
<i>cfg_type</i>	Type of configuration which will be fixed
<i>cfg_ver</i>	Version of configuration

#### Returns

Nothing

#### 4.3.2.2 [load\\_configuration\(\)](#)

```
void load_configuration (
    enum e\_cfg\_type cfg_type )
```

Load configuration specified by parameter `cfg_type`.

**Parameters**

<i>cfg_type</i>	Type of configuration which will be loaded
-----------------	--

**Returns**

Nothing

**4.3.2.3 save\_configuration()**

```
void save_configuration (
    enum e_cfg_type cfg_type )
```

Save configuration specified by parameter *cfg\_type*.

**Parameters**

<i>cfg_type</i>	Type of configuration which will be saved
-----------------	---

**Returns**

Nothing

**4.3.2.4 update\_configuration()**

```
void update_configuration (
    void )
```

Update all device configurations.

**Returns**

Nothing

**4.4 src/comm\_test.c File Reference**

```
#include "comm_test.h"
#include "main.h"
#include "pindef.h"
#include "config.h"
#include "extcomm.h"
#include "modules.h"
#include <pck.h>
#include <stdlib.h>
#include <globalrqstat.h>
#include <packet_handle.h>
```

## Functions

- void [ctest\\_answer\\_detected](#) (TPacket \*pck)  
*Checking if detected answer is correct.*
- void [ctest\\_answer\\_tout](#) (void)  
*Checking if response on send answer timeouts.*
- void [ctest\\_data\\_generate](#) (unsigned char \*buffer, unsigned char len)  
*Generate random data pattern of specified length.*
- void [ctest\\_send](#) (void)  
*Send packet for communication test.*
- void [ctest\\_start](#) (CommTestSettStruct \*set)  
*Start communication test.*
- void [ctest\\_stop](#) (void)  
*Stop active communication test.*
- void [ctest\\_get\\_result](#) (unsigned char \*buff)  
*Write communication test results to buffer.*
- void [ctest\\_handle](#) (void)  
*Handling active communication test.*

## Variables

- [CommTestSettStruct](#) **ctest\_settings**
- [CommTestResultStruct](#) **ctest**
- unsigned char **answer\_waiting**
- unsigned char **data** [MAX\_PACKET\_DATA\_SIZE]
- volatile unsigned char **id**

### 4.4.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.4.2 Function Documentation

#### 4.4.2.1 ctest\_answer\_detected()

```
void ctest_answer_detected (
    TPacket * pck )
```

Checking if detected answer is correct.

#### Parameters

in	<i>pck</i>	Pointer to received packet
----	------------	----------------------------



**Returns**

Nothing

**4.4.2.2 ctest\_answer\_tout()**

```
void ctest_answer_tout (
    void )
```

Checking if response on send answer timeouts.

**Returns**

Nothing

**4.4.2.3 ctest\_data\_generate()**

```
void ctest_data_generate (
    unsigned char * buffer,
    unsigned char len )
```

Generate random data pattern of specified length.

**Parameters**

<i>buffer</i>	Pointer to buffer for saving data
<i>len</i>	Generated data length

**Returns**

Nothing

**4.4.2.4 ctest\_get\_result()**

```
void ctest_get_result (
    unsigned char * buff )
```

Write communication test results to buffer.

**Parameters**

<i>out</i>	<i>buff</i>	Buffer for writing communication test result
------------	-------------	--

**Returns**

Nothing

**4.4.2.5 ctest\_handle()**

```
void ctest_handle (
    void )
```

Handling active communication test.

**Returns**

Nothing

**4.4.2.6 ctest\_send()**

```
void ctest_send (
    void )
```

Send packet for communication test.

**Returns**

Nothing

**4.4.2.7 ctest\_start()**

```
void ctest_start (
    CommTestSettStruct * set )
```

Start communication test.

**Parameters**

in	set	Pointer to settings of communication test
----	-----	---

**Returns**

Nothing

#### 4.4.2.8 ctest\_stop()

```
void ctest_stop (  
    void )
```

Stop active communication test.

##### Returns

Nothing

## 4.5 src/comm\_test.h File Reference

Testing communication between two devices via wireless network.

```
#include <pck.h>
```

### Classes

- struct [CommTestSettStruct](#)  
*Settings of communication test.*
- struct [CommTestResultStruct](#)  
*Result of communication test.*

### Functions

- void [ctest\\_answer\\_detected](#) (TPacket \*pck)  
*Checking if detected answer is correct.*
- void [ctest\\_answer\\_tout](#) (void)  
*Checking if response on send answer timeouts.*
- void [ctest\\_stop](#) (void)  
*Stop active communication test.*
- void [ctest\\_start](#) ([CommTestSettStruct](#) \*set)  
*Start communication test.*
- void [ctest\\_get\\_result](#) (unsigned char \*buff)  
*Write communication test results to buffer.*
- void [ctest\\_handle](#) (void)  
*Handling active communication test.*

### Variables

- [CommTestSettStruct](#) **ctest\_settings**
- [CommTestResultStruct](#) **ctest**

### 4.5.1 Detailed Description

Testing communication between two devices via wireless network.

Author

Ondrej Jerabek

### 4.5.2 Function Documentation

#### 4.5.2.1 ctest\_answer\_detected()

```
void ctest_answer_detected (
    TPacket * pck )
```

Checking if detected answer is correct.

Parameters

in	<i>pck</i>	Pointer to received packet
----	------------	----------------------------

Returns

Nothing

#### 4.5.2.2 ctest\_answer\_tout()

```
void ctest_answer_tout (
    void )
```

Checking if response on send answer timeouts.

Returns

Nothing

#### 4.5.2.3 ctest\_get\_result()

```
void ctest_get_result (
    unsigned char * buff )
```

Write communication test results to buffer.

**Parameters**

out	buff	Buffer for writing communication test result
-----	------	--

**Returns**

Nothing

**4.5.2.4 ctest\_handle()**

```
void ctest_handle (  
    void )
```

Handling active communication test.

**Returns**

Nothing

**4.5.2.5 ctest\_start()**

```
void ctest_start (  
    CommTestSettStruct * set )
```

Start communication test.

**Parameters**

in	set	Pointer to settings of communication test
----	-----	---

**Returns**

Nothing

**4.5.2.6 ctest\_stop()**

```
void ctest_stop (  
    void )
```

Stop active communication test.

**Returns**

Nothing

## 4.6 src/config.h File Reference

Configuration and macro definitions for libraries.

```
#include "eeprom.h"
#include "pindef.h"
#include "interrupt.h"
```

### Macros

- **#define USE\_CUTTER\_X**
- **#define CORE\_MEMORY\_EEPROM**
- **#define DEVICE\_TYPE** ARFID\_ETHERNET\_READER
- **#define FW\_VERSION\_MAJ** 1
- **#define FW\_VERSION\_MIN** 2
- **#define FW\_VERSION\_PATCH** 0
- **#define FW\_VERSION\_SUFFIX**
- **#define FW\_VERSION** FW v%d.%02d
- **#define HW\_VERSION** HW v%d.%02d
- **#define INFO\_SIZE** 50
- **#define SYS\_CLK** 96000000
- **#define CRYSTAL\_FREQ** 19200000UL
- **#define RTC\_CRYSTAL\_FREQ** 32768UL
- **#define MS\_TICK** (SystemCoreClock / 1000)
- **#define MAX\_PACKET\_DATA\_SIZE** 256
- **#define USE\_STD\_EEPROM**
- **#define STD\_EEPROM\_SIZE** 4032
- **#define LOG\_EEPROM\_ADDR** 0x00C0
- **#define LOG\_SIZE** 20
- **#define CFG\_EEPROM\_ADDR** 0x0020
- **#define CFG\_MODULE\_ADDR** 0x0100
- **#define CFG\_MODULE2\_OFFSET** 0x0100
- **#define CFG\_SX\_OFFSET** 0x0000
- **#define CFG\_NTR\_OFFSET** 0x0055
- **#define CFG\_DW\_OFFSET** 0x00AA
- **#define EXTFLASH\_SIZE** 2097152
- **#define USE\_EXT\_FLASH**
- **#define SYSTEM\_EXT**
- **#define SYSTEM\_EXT\_USE\_EXTFLASH**
- **#define FLASHING\_BOTTOM** 0
- **#define EXTFLASH\_WAIT\_FOR\_WRITE** 1
- **#define EXTFLASH\_WAIT\_FOR\_ERASE** 0
- **#define FLASHING\_TOP** 524288
- **#define USE\_SPI\_WRAPPER**
- **#define FLASH\_SPI** 0
- **#define EEPROM\_SPI** 0
- **#define MODULES\_SPI** 1
- **#define NTRX\_SPI** 1
- **#define SPI\_0\_FREQ** 10000000
- **#define SX\_SPI1** 1
- **#define SX\_SPI2** 1
- **#define SPI\_1\_FREQ** 1000000

- `#define NO_MULTIMASTER_MODE`
- `#define UART_SYS_BAUD_RATE 57600`
- `#define UART_SYS 3`
- `#define USE_UART_3`
- `#define UART_3_RXBUFFSIZE 275`
- `#define UART_3_TXBUFFSIZE 275`
- `#define UART_3_TX(x)`
- `#define UART_3_RX(x)`
- `#define UART_3_USE_TXEN 0`
- `#define UART_TX_TIMEOUT 1`
- `#define UART_NEXT_PCK_TIMEOUT 2`
- `#define UART_RESP_TIMEOUT 1`
- `#define UART_REQ_TIMEOUT 2`
- `#define PRINTF_UART UART_SYS`
- `#define UART_PRINTF`
- `#define COMM_ISR_FREQ`
- `#define RF_FREQUENCY_1 868300000`
- `#define RF_FREQUENCY_2 867300000`
- `#define RF_BAUD`
- `#define SX_XTAL_FREQ 11059200`
- `#define VCO_TRIM VCO_TRIM_60`
- `#define SX_USE_CRC`
- `#define SYNCWORD`
- `#define SYNCWORD_LENGTH`
- `#define SX_NUMBER 2`
- `#define USE_SX_RSSI`
- `#define SX_CLKOUT CLKOUT_ENABLED`
- `#define NO_SX_PCK_ID_GEN`
- `#define RTC_TICK_FREQ 1000`
- `#define RTC_SYNC_TIMECONST 100`
- `#define RTC_MAX_DIFF 5`
- `#define RTC_SYNC_INTERVAL 10000`
- `#define ETH_PHY LAN8720`
- `#define NO_TCP_PCK_SEND_RECEIVE`
- `#define TCPSTACK_CONFIG_OFFSET 0x0080`
- `#define APPS_NAME "Arfid Ethernet Reader"`
- `#define MY_DEFAULT_HOST_NAME "arfid"`
- `#define HTTP_NAME_PASS`
- `#define PASW`
- `#define LWIP_UDP_CUTTER_TX 0`
- `#define LWIP_UDP_CUTTER_TX2 1`
- `#define USE_BOOTLOADER`
- `#define BOOTLOADER_MACRO tmr.bootloader = 200`
- `#define DEV_HANDLER_NEW`
- `#define ANSWER_HANDLER`
- `#define DEV_BROADCAST_HANDLER_NEW`
- `#define NO_PCK_ID_GEN`
- `#define ROUTING_TABLE_ENABLED`
- `#define ANSWER_TIMEOUT 100`
- `#define RSSI_TABLE_SIZE 100`
- `#define RSSI_TIMEOUT 10`
- `#define TIMESYNC_BROADCAST_TIMEOUT 1500`
- `#define BATT_LOW_VOLTAGE 3250`
- `#define BATT_CRITICAL_VOLTAGE 3100`
- `#define BTN_POWER_OFF_TIME 5000`

- #define **BTN\_RESET\_TIME** 500
- #define **TMR\_LED** 100
- #define **AVG\_SAMPLES** 5
- #define **V\_BATT\_CHNL** ADC\_CH0
- #define **V\_SUPPLY\_CHNL** ADC\_CH1
- #define **V\_USB\_CHNL** ADC\_CH6
- #define **I\_MEAS\_CHNL** ADC\_CH7
- #define **DIV\_BATT\_R1** 560.0
- #define **DIV\_BATT\_R2** 560.0
- #define **DIV\_USB\_R1** 20.0
- #define **DIV\_USB\_R2** 10.0
- #define **DIV\_SUPPLY\_R1** 200.0
- #define **DIV\_SUPPLY\_R2** 10.0
- #define **R\_ISET** 1500.
- #define **ADC\_LEVELS** 4096.
- #define **VREF** 3000.
- #define **DIVIDER**(R1, R2) (VREF / (ADC\_LEVELS \* (R2 / (R1 + R2))))
- #define **DS\_TREG\_CHECK**
- #define **DS\_DI**
- #define **ds\_di**() di()
- #define **ds\_ei**() ei()
- #define **ds\_dil**() di()
- #define **ds\_eil**() ei()
- #define **DS\_TREG\_CHECK**
- #define **DS\_DI**
- #define **ds\_di**() di()
- #define **ds\_ei**() ei()
- #define **ds\_dil**() di()
- #define **ds\_eil**() ei()
- #define **dqz**() DS\_DIR(0)
- #define **dqhigh**() DS\_DIR(1), DS\_OUT(1)
- #define **dqlow**() DS\_DIR(1), DS\_OUT(0)
- #define **dq**() DS\_IN
- #define **CONFIG\_NTRX\_RECAL\_DELAY** (3000)
- #define **CONFIG\_NTRX\_PHY\_RX\_TIMEOUT** (5000)
- #define **CONFIG\_TRAFFIC\_LED** 1
- #define **CONFIG\_USE\_TX\_LED** 1
- #define **CONFIG\_USE\_RX\_LED** 1
- #define **CONFIG\_DEFAULT\_SYNCWORD**
- #define **CONFIG\_NTRX\_22MHZ\_4000NS** 0x03
- #define **CONFIG\_NTRX\_80MHZ\_1000NS** 0x06
- #define **CONFIG\_NTRX\_80MHZ\_2000NS** 0x07
- #define **CONFIG\_NTRX\_80MHZ\_4000NS** 0x08
- #define **CONFIG\_NTRX\_22MHZ\_4000NS\_ROM** 0x0A
- #define **CONFIG\_DEFAULT\_MODE** CONFIG\_NTRX\_80MHZ\_4000NS
- #define **CONFIG\_MAX\_ARQ** 1
- #define **CONFIG\_MAX\_LOG\_CHANNEL** 4
- #define **CONFIG\_MAX\_PACKET\_SIZE** 128
- #define **NTRXRANGING\_SELECT\_TYPES**
- #define **SUPPORT\_RANGING\_TYPE\_3W\_A**
- #define **NTRX\_COUNT** 2
- #define **USE\_USB\_PLL**
- #define **USB\_DEVICE\_ONLY**
- #define **USB\_UP\_LED**
- #define **USB\_SOFT\_CONNECT**
- #define **USB\_SOFT\_CONNECT\_TIMER** 3000
- #define **PRODUCT\_STRING** "Ethernet Reader"



### 4.6.1 Detailed Description

Configuration and macro definitions for libraries.

Author

Ondrej Jerabek

Main configuration and definitions

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 USE\_USB\_PLL

```
#define USE_USB_PLL
```

LPCUSBLib: Microcontroller architecture: ARCH\_LPC (Chip architecture. Must be ARCH\_LPC) Microcontroller model: **LPC11UXX, LPC13UXX, LPC17XX, LPC177X\_8X, LPC18XX, LPC43XX** (see USBMode.h) nxpUSBLib mode configuration: USB\_DEVICE\_ONLY (Build only device related code in the USB library), USB\_HOST\_ONLY (Build only host related code in the USB library)

USB\_FORCED\_FULLSPEED 1/0: 1 - operate ALWAYS in USB Full-Speed (12 Mbit/s) although some MCU can support USB Hi-Speed (480 Mbit/s), 0 - use max supported USB speed USE\_USB\_ROM\_STACK 1/0: 1 - use MCU's internal ROM USB stack (only some MCU), 0 - no USB ROM stack USB\_PORT 0/1/0xFF: 0 - use USB0, 1 - use USB1, 0xFF - MultiPort (all (or rather both) supported USB peripherals)

## 4.7 src/extcomm.c File Reference

```
#include "main.h"
#include "pindef.h"
#include "config.h"
#include "extcomm.h"
#include "cfg.h"
#include "modules.h"
#include "routing.h"
#include "comm_test.h"
#include "rssi.h"
#include <stdio.h>
#include <simplelpcio.h>
#include <system.h>
#include <eeprom.h>
#include <delay.h>
#include <sx_rssi.h>
#include <packet_handle.h>
#include <interrupt.h>
#include <multiuart.h>
#include <multiuart_packet.h>
#include <usb_packet.h>
#include <udp_packet.h>
#include <tcp_packet.h>
#include <tcplink.h>
#include <cdc_vcom.h>
#include <wdt.h>
```

## Macros

- #define **ANSWER\_COND**(s) (s == PCK\_OK)

## Functions

- void **answer\_ms\_tick** (void)  
*Handling packet answer each millisecond.*
- void **set\_answer\_timer** (unsigned long timeout)  
*Set and start answer timer.*
- void **clear\_answer\_timer** (void)  
*Stop and clear answer timer.*
- void **answer\_timeout\_handle** (void)  
*Handle answer waiting timeout.*
- enum e\_pck\_handle\_status **sx1\_pck\_handle** (void)  
*Handle packets incoming through semtech module 1.*
- enum e\_pck\_handle\_status **sx2\_pck\_handle** (void)  
*Handle packets incoming through semtech module 2.*
- enum e\_pck\_handle\_status **uart\_sys\_pck\_handle** (void)  
*Handling packets from system uart.*
- void **module1\_handle** (void)  
*Module 1 packets and communication handling.*
- void **module2\_handle** (void)  
*Module 2 packets and communication handling.*
- enum e\_pck\_handle\_status **usb\_pck\_handle** (void)  
*Handling packets from USB.*
- enum e\_pck\_handle\_status **tcp\_pck\_handle** (unsigned char channel)  
*Handling packets from TCP.*
- enum e\_pck\_handle\_status **udp\_pck\_handle** (unsigned char channel)  
*Handling packets from UDP.*
- bool **send\_wakeup\_burst** (Addr\_t src\_addr, Addr\_t dest\_addr, unsigned short count, unsigned char rq, unsigned char data\_len, unsigned char \*data)  
*Sending wakeup burst.*
- enum e\_pck\_handle\_status **is\_dev\_handle** (enum e\_packet\_source\_device source\_device, enum e\_pck\_handle\_status receive\_status, TPacket \*pck)  
*Internal Service device requests handling.*
- enum e\_pck\_handle\_status **dev\_handle** (enum e\_packet\_source\_device source\_device, enum e\_pck\_handle\_status receive\_status, TPacket \*pck)  
*Device request packet handling.*
- enum e\_pck\_handle\_status **dev\_broadcast\_handle** (enum e\_packet\_source\_device source\_device, enum e\_pck\_handle\_status receive\_status, TPacket pck\_in, TPacket \*pck\_out)  
*Broadcast packet handling.*
- enum e\_pck\_handle\_status **answer\_handle** (enum e\_packet\_source\_device source\_device, enum e\_pck\_handle\_status receive\_status, TPacket \*pck)  
*Answer packet handling.*
- enum e\_pck\_handle\_status **packet\_route** (enum e\_packet\_source\_device source\_device, TPacket packet\_in, TPacket \*packet\_out)  
*Routing packet handling.*
- void **beacon\_broadcast** (void)  
*Sending beacon broadcast packets.*

## Variables

- unsigned char **outbuff** [MAX\_PACKET\_DATA\_SIZE]
- [CommStruct](#) **comm**

### 4.7.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.7.2 Function Documentation

#### 4.7.2.1 answer\_handle()

```
enum e_pck_handle_status answer_handle (
    enum e_packet_source_device source_device,
    enum e_pck_handle_status receive_status,
    TPacket * pck )
```

Answer packet handling.

#### Parameters

	<i>source_device</i>	Device peripheral which received packet (USB, UART, etc.)
	<i>receive_status</i>	Status of received packet
in	<i>pck</i>	Pointer to incoming packet

#### Returns

Result of packet processing

#### Note

Called only if device waiting for response to packet

#### 4.7.2.2 answer\_ms\_tick()

```
void answer_ms_tick (
    void )
```

Handling packet answer each millisecond.

**Returns**

Nothing

**Note**

Should be called each millisecond in SysTick interrupt

**4.7.2.3 answer\_timeout\_handle()**

```
void answer_timeout_handle (
    void )
```

Handle answer waiting timeout.

**Returns**

Nothing

**4.7.2.4 beacon\_broadcast()**

```
void beacon_broadcast (
    void )
```

Sending beacon broadcast packets.

**Returns**

Nothing

**4.7.2.5 clear\_answer\_timer()**

```
void clear_answer_timer (
    void )
```

Stop and clear answer timer.

**Returns**

Nothing

**4.7.2.6 dev\_broadcast\_handle()**

```
enum e_pck_handle_status dev_broadcast_handle (
    enum e_packet_source_device source_device,
    enum e_pck_handle_status receive_status,
    TPacket pck_in,
    TPacket * pck_out )
```

Broadcast packet handling.

**Parameters**

	<i>source_device</i>	Device peripheral which received packet (USB, UART, etc.)
	<i>receive_status</i>	Status of received packet
	<i>pck_in</i>	Incoming packet
out	<i>pck_out</i>	Pointer to Outcoming packet buffer

**Returns**

Result of packet processing

**Note**

Called from library after broadcast packet reception

**4.7.2.7 dev\_handle()**

```
enum e_pck_handle_status dev_handle (
    enum e_packet_source_device source_device,
    enum e_pck_handle_status receive_status,
    TPacket * pck )
```

Device request packet handling.

**Parameters**

	<i>source_device</i>	Device peripheral which received packet (USB, UART, etc.)
	<i>receive_status</i>	Status of received packet
in, out	<i>pck</i>	Pointer to received packet. Used for answer packet too

**Returns**

Result of packet processing

**Note**

Called from library after packet reception and when destination address equal with device address

**4.7.2.8 is\_dev\_handle()**

```
enum e_pck_handle_status is_dev_handle (
    enum e_packet_source_device source_device,
    enum e_pck_handle_status receive_status,
    TPacket * pck )
```

Internal Service device requests handling.

**Parameters**

	<i>source_device</i>	Device peripheral which received packet (USB, UART, etc.)
	<i>receive_status</i>	Status of received packet
<i>in, out</i>	<i>pck</i>	Pointer to received packet. Used for answer packet too

**Returns**

Result of packet processing

**4.7.2.9 module1\_handle()**

```
void module1_handle (  
    void )
```

Module 1 packets and communication handling.

**Returns**

Nothing

**4.7.2.10 module2\_handle()**

```
void module2_handle (  
    void )
```

Module 2 packets and communication handling.

**Returns**

Nothing

**4.7.2.11 packet\_route()**

```
enum e_pck_handle_status packet_route (  
    enum e_packet_source_device source_device,  
    TPacket packet_in,  
    TPacket * packet_out )
```

Routing packet handling.

## Parameters

	<i>source_device</i>	Device peripheral which received packet (USB, UART, etc.)
	<i>pck_in</i>	Incoming packet
out	<i>pck_out</i>	Pointer to Outcoming packet buffer

## Returns

Result of packet processing

## Note

Called from library after packet reception and when destination address not equal with device address

## 4.7.2.12 send\_wakeup\_burst()

```
bool send_wakeup_burst (
    Addr_t src_addr,
    Addr_t dest_addr,
    unsigned short count,
    unsigned char rq,
    unsigned char data_len,
    unsigned char * data )
```

Sending wakeup burst.

## Parameters

	<i>src_addr</i>	Address of source device
	<i>dest_addr</i>	Address of destination device
	<i>count</i>	Number of packet in burst
	<i>rq</i>	Request send in packets
	<i>data_len</i>	Length of data send in packet
in	<i>data</i>	Pointer to data

## Returns

`true` if packet successfully send

## 4.7.2.13 set\_answer\_timer()

```
void set_answer_timer (
    unsigned long timeout )
```

Set and start answer timer.

**Parameters**

<i>timeout</i>	Answer timeout in milliseconds
----------------	--------------------------------

**Returns**

Nothing

**4.7.2.14 `sx1_pck_handle()`**

```
enum e_pck_handle_status sx1_pck_handle (  
    void )
```

Handle packets incoming through semtech module 1.

**Returns**

Nothing

**4.7.2.15 `sx2_pck_handle()`**

```
enum e_pck_handle_status sx2_pck_handle (  
    void )
```

Handle packets incoming through semtech module 2.

**Returns**

Nothing

**4.7.2.16 `tcp_pck_handle()`**

```
enum e_pck_handle_status tcp_pck_handle (  
    unsigned char channel )
```

Handling packets from TCP.

**Returns**

Result of packet processing



#### 4.7.2.17 uart\_sys\_pck\_handle()

```
enum e_pck_handle_status uart_sys_pck_handle (
    void )
```

Handling packets from system uart.

##### Returns

Result of packet processing

#### 4.7.2.18 udp\_pck\_handle()

```
enum e_pck_handle_status udp_pck_handle (
    unsigned char channel )
```

Handling packets from UDP.

##### Returns

Result of packet processing

#### 4.7.2.19 usb\_pck\_handle()

```
enum e_pck_handle_status usb_pck_handle (
    void )
```

Handling packets from USB.

##### Returns

Result of packet processing

## 4.8 src/extcomm.h File Reference

Extended communication functions.

```
#include <pck.h>
```

### Classes

- struct [CommStruct](#)  
*Parameters for answer detection.*

## Functions

- enum e\_pck\_handle\_status [uart\\_sys\\_pck\\_handle](#) (void)  
*Handling packets from system uart.*
- void [module1\\_handle](#) (void)  
*Module 1 packets and communication handling.*
- void [module2\\_handle](#) (void)  
*Module 2 packets and communication handling.*
- enum e\_pck\_handle\_status [usb\\_pck\\_handle](#) (void)  
*Handling packets from USB.*
- enum e\_pck\_handle\_status [tcp\\_pck\\_handle](#) (unsigned char channel)  
*Handling packets from TCP.*
- enum e\_pck\_handle\_status [udp\\_pck\\_handle](#) (unsigned char channel)  
*Handling packets from UDP.*
- bool [send\\_wakeup\\_burst](#) (Addr\_t src\_addr, Addr\_t dest\_addr, unsigned short count, unsigned char rq, unsigned char data\_len, unsigned char \*data)  
*Sending wakeup burst.*
- void [answer\\_ms\\_tick](#) (void)  
*Handling packet answer each millisecond.*
- void [answer\\_timeout\\_handle](#) (void)  
*Handle answer waiting timeout.*
- void [set\\_answer\\_timer](#) (unsigned long timeout)  
*Set and start answer timer.*
- void [clear\\_answer\\_timer](#) (void)  
*Stop and clear answer timer.*
- void [beacon\\_broadcast](#) (void)  
*Sending beacon broadcast packets.*

## Variables

- unsigned char **outbuff** [MAX\_PACKET\_DATA\_SIZE]
- [CommStruct](#) **comm**

### 4.8.1 Detailed Description

Extended communication functions.

#### Author

Ondrej Jerabek

### 4.8.2 Function Documentation

#### 4.8.2.1 answer\_ms\_tick()

```
void answer_ms_tick (  
    void )
```

Handling packet answer each millisecond.

##### Returns

Nothing

##### Note

Should be called each millisecond in SysTick interrupt

#### 4.8.2.2 answer\_timeout\_handle()

```
void answer_timeout_handle (  
    void )
```

Handle answer waiting timeout.

##### Returns

Nothing

#### 4.8.2.3 beacon\_broadcast()

```
void beacon_broadcast (  
    void )
```

Sending beacon broadcast packets.

##### Returns

Nothing

#### 4.8.2.4 clear\_answer\_timer()

```
void clear_answer_timer (  
    void )
```

Stop and clear answer timer.

##### Returns

Nothing

#### 4.8.2.5 module1\_handle()

```
void module1_handle (
    void )
```

Module 1 packets and communication handling.

##### Returns

Nothing

#### 4.8.2.6 module2\_handle()

```
void module2_handle (
    void )
```

Module 2 packets and communication handling.

##### Returns

Nothing

#### 4.8.2.7 send\_wakeup\_burst()

```
bool send_wakeup_burst (
    Addr_t src_addr,
    Addr_t dest_addr,
    unsigned short count,
    unsigned char rq,
    unsigned char data_len,
    unsigned char * data )
```

Sending wakeup burst.

##### Parameters

	<i>src_addr</i>	Address of source device
	<i>dest_addr</i>	Address of destination device
	<i>count</i>	Number of packet in burst
	<i>rq</i>	Request send in packets
	<i>data_len</i>	Length of data send in packet
in	<i>data</i>	Pointer to data

**Returns**

true if packet successfully send

**4.8.2.8 set\_answer\_timer()**

```
void set_answer_timer (
    unsigned long timeout )
```

Set and start answer timer.

**Parameters**

<i>timeout</i>	Answer timeout in milliseconds
----------------	--------------------------------

**Returns**

Nothing

**4.8.2.9 tcp\_pck\_handle()**

```
enum e_pck_handle_status tcp_pck_handle (
    unsigned char channel )
```

Handling packets from TCP.

**Returns**

Result of packet processing

**4.8.2.10 uart\_sys\_pck\_handle()**

```
enum e_pck_handle_status uart_sys_pck_handle (
    void )
```

Handling packets from system uart.

**Returns**

Result of packet processing

#### 4.8.2.11 udp\_pck\_handle()

```
enum e_pck_handle_status udp_pck_handle (
    unsigned char channel )
```

Handling packets from UDP.

##### Returns

Result of packet processing

#### 4.8.2.12 usb\_pck\_handle()

```
enum e_pck_handle_status usb_pck_handle (
    void )
```

Handling packets from USB.

##### Returns

Result of packet processing

## 4.9 src/hardfault.c File Reference

### 4.9.1 Detailed Description

#### Author

Ondrej Jerabek

## 4.10 src/main.c File Reference

```
#include <stdio.h>
#include "ARM_Ethernet_Reader.h"
#include "config.h"
#include "main.h"
#include "power_ctrl.h"
#include "modules.h"
#include "cfg.h"
#include "spi_wrapper.h"
#include "extcomm.h"
#include "comm_test.h"
#include "rssi.h"
#include "chip.h"
#include "eeprom.h"
#include "gpio_17xx_40xx.h"
#include "enet_17xx_40xx.h"
#include <coremem.h>
```

```

#include <system.h>
#include <run_bootloader.h>
#include <device_types.h>
#include <interrupt.h>
#include <cr_section_macros.h>
#include <dallas.h>
#include <watchdog.h>
#include <cdc_main.h>
#include <cdc_vcom.h>
#include <multisx1211.h>
#include <multisx1211_packet.h>
#include <sx_rssi.h>
#include <multiuart.h>
#include <multiuart_packet.h>
#include <networking.h>
#include <udp_packet.h>
#include <phy/generic_phy.h>
#include <phy/mdio_wrapper.h>
#include <ds18b20.h>
#include <systemext.h>
#include <m25p80.h>
#include <adc.h>

```

## Functions

- void [SysTick\\_Handler](#) (void)  
*Handling millisecond system timer interrupt.*
- void [GPIO\\_IRQHandler](#) (void)  
*Handling interrupts from modules.*
- unsigned long [read\\_system\\_timer](#) (void)  
*Reading 32bit system timer.*
- void [SystemCoreClock\\_init](#) (void)  
*Initializing internal PLLs and core clock.*
- void [SystemClkOut\\_init](#) (bool enabled)  
*Initializing 32 MHz clock output for nanotron module.*
- void [Chip\\_USB\\_Init](#) (void)  
*Initializing clock for USB peripheral.*
- void [reinit\\_SX\\_modules](#) (void)  
*Reinitialization SX modules if someone unexpectedly restart.*
- void [ADC\\_init](#) (void)  
*Initializing ADC converter for measuring voltages and currents.*
- int [main](#) (void)  
*Main function of firmware.*

## Variables

- TCoreRAM **core**
- [EthernetReader\\_State](#) **st**
- [EthernetReader\\_Cfg](#) **cfg**
- [ModuleCfg](#) **module\_cfg** [2]
- [Flag\\_struct](#) **flags**
- [Timer\\_struct](#) **tmr**
- [Leds\\_struct](#) **led**
- unsigned char **info** [INFO\_SIZE]
- volatile uint16\_t **mdio\_result** = 0
- volatile uint16\_t **pruchody** = 0

### 4.10.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.10.2 Function Documentation

#### 4.10.2.1 ADC\_init()

```
void ADC_init (
    void )
```

Initializing ADC converter for measuring voltages and currents.

#### Returns

Nothing

#### Note

After initialization start converting in Burst mode

#### 4.10.2.2 Chip\_USB\_Init()

```
void Chip_USB_Init (
    void )
```

Initializing clock for USB peripheral.

#### Returns

Nothing

#### Note

Overrides clock initialization in USB library



#### 4.10.2.3 GPIO\_IRQHandler()

```
void GPIO_IRQHandler (
    void )
```

Handling interrupts from modules.

##### Returns

Nothing

#### 4.10.2.4 main()

```
int main (
    void )
```

Main function of firmware.

##### Returns

Ideally Nothing. Never ending function

#### 4.10.2.5 read\_system\_timer()

```
unsigned long read_system_timer (
    void )
```

Reading 32bit system timer.

##### Returns

Value of system timer in [ms]

#### 4.10.2.6 reinit\_SX\_modules()

```
void reinit_SX_modules (
    void )
```

Reinitialization SX modules if someone unexpectedly restart.

##### Returns

Nothing

#### 4.10.2.7 SystemClkOut\_init()

```
void SystemClkOut_init (
    bool enabled )
```

Initializing 32 MHz clock output for nanotron module.

**Parameters**

<i>enabled</i>	Enable clock output
----------------	---------------------

**Returns**

Nothing

**4.10.2.8 SystemCoreClock\_init()**

```
void SystemCoreClock_init (  
    void )
```

Initializing internal PLLs and core clock.

**Returns**

Nothing

**Note**

Overrides core clock initialization in library

**4.10.2.9 SysTick\_Handler()**

```
void SysTick_Handler (  
    void )
```

Handling millisecond system timer interrupt.

**Returns**

Nothing

**Note**

Called by SysTick interrupt

**4.11 src/main.h File Reference**

Function and structures defined in main program.

```
#include "config.h"  
#include "chip.h"  
#include "ARM_Ethernet_Reader.h"  
#include "coremem.h"
```

## Classes

- struct [Flag\\_struct](#)  
*Flags.*
- struct [Timer\\_struct](#)  
*Timers.*
- struct [Leds\\_struct](#)  
*Variables for LED lighting and blinking.*

## Macros

- #define [handle\\_led](#)(var, led)  
*Handling LED lighting.*
- #define [handle\\_timer](#)(timer)  
*Handling timer decrement.*
- #define [handle\\_timer\\_flags](#)(tmr, val, flag)  
*Handling timer decrement, loading and flag.*
- #define [handle\\_led\\_blink](#)(tmr, blink, invert, led)  
*Handling led blinking.*

## Functions

- unsigned long [read\\_system\\_timer](#) (void)  
*Reading 32bit system timer.*
- void [Chip\\_USB\\_Init](#) (void)  
*Initializing clock for USB peripheral.*
- void [SystemClkOut\\_init](#) (bool enabled)  
*Initializing 32 MHz clock output for nanotron module.*

## Variables

- unsigned char **info** [INFO\_SIZE]
- TCoreRAM **core**
- [Flag\\_struct](#) **flags**
- [Timer\\_struct](#) **tmr**
- [Leds\\_struct](#) **led**
- [EthernetReader\\_State](#) **st**
- [EthernetReader\\_Cfg](#) **cfg**
- [ModuleCfg](#) **module\_cfg** [2]

### 4.11.1 Detailed Description

Function and structures defined in main program.

#### Author

Ondrej Jerabek

## 4.11.2 Macro Definition Documentation

### 4.11.2.1 `handle_led`

```
#define handle_led(  
    var,  
    led )
```

#### Value:

```
{  
    if (var)  
    {  
        if (var != 0xffff)  
            var--;  
        led(1);  
    }  
    else  
        led(0);  
}
```

Handling LED lighting.

#### Parameters

<i>var</i>	LED lighting timer
<i>led</i>	LED pin definition

### 4.11.2.2 `handle_led_blink`

```
#define handle_led_blink(  
    tmr,  
    blink,  
    invert,  
    led )
```

#### Value:

```
{  
    if (tmr > 0)  
    {  
        tmr--;  
        if (tmr > (TMR_LED / 2))  
            led(!(invert));  
        else  
            led(invert);  
    }  
    else  
    {  
        led(invert);  
        if (blink)  
        {  
            blink = false;  
            tmr = TMR_LED;  
        }  
    }  
}
```

Handling led blinking.

**Parameters**

<i>tmr</i>	LED lighting timer
<i>blink</i>	Perform LED blink
<i>invert</i>	Inverts LED blinking
<i>led</i>	LED pin definition

**4.11.2.3 handle\_timer**

```
#define handle_timer(  
    timer )
```

**Value:**

```
{  
    if (timer)           \  
        timer--;        \  
}
```

Handling timer decrement.

**Parameters**

<i>timer</i>	Handled timer
--------------	---------------

**4.11.2.4 handle\_timer\_flags**

```
#define handle_timer_flags(  
    tmr,  
    val,  
    flag )
```

**Value:**

```
{  
    if (tmr)             \  
        tmr--;          \  
    else {               \  
        tmr = val;       \  
        flag = true;     \  
    }                   \  
}
```

Handling timer decrement, loading and flag.

**Parameters**

<i>tmr</i>	Handled timer
<i>val</i>	Value loaded to timer when reaches zero
<i>flag</i>	Flag which is set when timer reaches zero

### 4.11.3 Function Documentation

#### 4.11.3.1 Chip\_USB\_Init()

```
void Chip_USB_Init (
    void )
```

Initializing clock for USB peripheral.

##### Returns

Nothing

##### Note

Overrides clock initialization in USB library

#### 4.11.3.2 read\_system\_timer()

```
unsigned long read_system_timer (
    void )
```

Reading 32bit system timer.

##### Returns

Value of system timer in [ms]

#### 4.11.3.3 SystemClkOut\_init()

```
void SystemClkOut_init (
    bool enabled )
```

Initializing 32 MHz clock output for nanotron module.

##### Parameters

<i>enabled</i>	Enable clock output
----------------	---------------------

**Returns**

Nothing

## 4.12 src/main\_bootloader.c File Reference

```
#include <stdio.h>
#include "config.h"
#include "main.h"
#include "bootloader.h"
#include "spi_wrapper.h"
#include "chip.h"
```

**Functions**

- int [main](#) (void)

*Main function of bootloader.*

### 4.12.1 Detailed Description

**Author**

Ondrej Jerabek

### 4.12.2 Function Documentation

#### 4.12.2.1 main()

```
int main (
    void )
```

Main function of bootloader.

**Returns**

Zero (0)

**Note**

Running main firmware or copying new firmware to MCU



## 4.13 src/modules.c File Reference

```
#include "modules.h"
#include "rssi.h"
#include "type.h"
#include "nanotron.h"
#include <main.h>
#include <sx_rssi.h>
#include <spi.h>
#include <delay.h>
#include <pck.h>
#include <sx1211config.h>
#include <multisx1211.h>
#include <multisx1211_packet.h>
#include <ntrx_packet.h>
```

### Macros

- #define **SPI\_INDEX** MODULES\_SPI
- #define **sx\_select**(module)  
*Selecting semtech module on SPI bus.*
- #define **sx\_unselect**()  
*Unselecting semtech modules on SPI bus.*
- #define **dw\_select**(module)  
*Selecting decawave module on SPI bus.*
- #define **dw\_unselect**()  
*Unselecting decawave modules on SPI bus.*
- #define **ntr\_select**(module)  
*Selecting nanotron module on SPI bus.*
- #define **ntr\_unselect**()  
*Unselecting nanotron modules on SPI bus.*
- #define **module\_reset**(module, state)  
*Set reset pin on module to defined state.*

### Functions

- void **module\_pck\_send** (enum **e\_module** module, TPacket \*pck)  
*Sending packet through selected module.*
- enum e\_send\_receive\_status **module\_pck\_send\_receive** (enum **e\_module** module, TPacket \*pckin, TPacket \*pckout, unsigned char timeout)  
*Sending and receiving packet through selected module.*
- unsigned char **module\_get\_rssi** (enum **e\_module** module)  
*Measuring Receive Signal Strength Intensity of received packet.*
- void **module\_set\_sleep** (enum **e\_module** module)  
*Set selected module to sleep mode.*
- unsigned char **module\_get\_rssi\_background** (enum **e\_module** module)  
*Measuring Receive Signal Strength Intensity of background.*
- void **module\_set\_pwr** (enum **e\_module** module, unsigned char pwr\_state)  
*Switching power to each module.*
- bool **module\_sx\_present** (enum **e\_module** module)

- *Semtech module detection.*  
bool `module_dw_present` (enum `e_module` module)
- *Decawave module detection.*  
bool `module_ntr_present` (enum `e_module` module)
- *Nanotron module detection.*  
void `module_detect` (enum `e_module` module)
- *Detect type of module plugged into slot.*  
void `module_configure_pins` (enum `e_module` module, enum `e_module_type` type)
- *Set correct pin configuration for specified module type.*  
void `modules_init` (void)
- *Initialize all modules in slots.*

## Variables

- `MODULE_STRUCT` modules

### 4.13.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.13.2 Macro Definition Documentation

#### 4.13.2.1 `dw_select`

```
#define dw_select(  
    module )
```

#### Value:

```
{  
    if (module == MODULE_1)           \  
        DW_CS_0(0);                 \  
    else if (module == MODULE_2)     \  
        DW_CS_1(0);                 \  
}
```

Selecting decawave module on SPI bus.

#### Parameters

<code>module</code>	Module index
---------------------	--------------

## 4.13.2.2 dw\_unselect

```
#define dw_unselect( )
```

**Value:**

```
{
    DW_CS_0(1);
    DW_CS_1(1);
}
```

Unselecting decawave modules on SPI bus.

## 4.13.2.3 module\_reset

```
#define module_reset(
    module,
    state )
```

**Value:**

```
{
    if (module == MODULE_1)
        MODULE1_RESET(state);
    else if (module == MODULE_2)
        MODULE2_RESET(state);
}
```

Set reset pin on module to defined state.

**Parameters**

<i>module</i>	Module index
<i>state</i>	Reset pin state

## 4.13.2.4 ntr\_select

```
#define ntr_select(
    module )
```

**Value:**

```
{
    if (module == MODULE_1)
        NTRX_1_CS(0);
    else if (module == MODULE_2)
        NTRX_2_CS(0);
}
```

Selecting nanotron module on SPI bus.

## Parameters

<i>module</i>	Module index
---------------	--------------

## 4.13.2.5 ntr\_unselect

```
#define ntr_unselect( )
```

## Value:

```
{
    NTRX_1_CS(1);
    NTRX_2_CS(1);
}
```

Unselecting nanotron modules on SPI bus.

## 4.13.2.6 sx\_select

```
#define sx_select(
    module )
```

## Value:

```
{
    if (module == MODULE_1)
        SX_CONFIG_SELECT1();
    else if (module == MODULE_2)
        SX_CONFIG_SELECT2();
}
```

Selecting semtech module on SPI bus.

## Parameters

<i>module</i>	Module index
---------------	--------------

## 4.13.2.7 sx\_unselect

```
#define sx_unselect( )
```

## Value:

```
{
    SX_CONFIG_UNSELECT1();
    SX_CONFIG_UNSELECT2();
}
```

Unselecting semtech modules on SPI bus.

### 4.13.3 Function Documentation

#### 4.13.3.1 module\_configure\_pins()

```
void module_configure_pins (
    enum e_module module,
    enum e_module_type type )
```

Set correct pin configuration for specified module type.

##### Parameters

<i>module</i>	Module index
<i>type</i>	Type of module plugged into slot

##### Returns

Nothing

#### 4.13.3.2 module\_detect()

```
void module_detect (
    enum e_module module )
```

Detect type of module plugged into slot.

##### Parameters

<i>module</i>	Module index
---------------	--------------

##### Returns

Nothing

#### 4.13.3.3 module\_dw\_present()

```
bool module_dw_present (
    enum e_module module )
```

Decawave module detection.

##### Parameters

<i>module</i>	Module index
---------------	--------------

##### Returns

true if module was detected

#### 4.13.3.4 module\_get\_rssi()

```
unsigned char module_get_rssi (
    enum e_module module )
```

Measuring Receive Signal Strength Intensity of received packet.

##### Parameters

<i>module</i>	Module index
---------------	--------------

##### Returns

Received signal strength

##### Note

Implemented only on Semtech module

#### 4.13.3.5 module\_get\_rssi\_background()

```
unsigned char module_get_rssi_background (
    enum e_module module )
```

Measuring Receive Signal Strength Intensity of background.

##### Parameters

<i>module</i>	Module index
---------------	--------------

**Returns**

Received signal strength

**Note**

Implemented only on Semtech module and Nanotron module

**4.13.3.6 module\_ntr\_present()**

```
bool module_ntr_present (
    enum e_module module )
```

Nanotron module detection.

**Parameters**

<i>module</i>	Module index
---------------	--------------

**Returns**

true if module was detected

**4.13.3.7 module\_pck\_send()**

```
void module_pck_send (
    enum e_module module,
    TPacket * pck )
```

Sending packet through selected module.

**Parameters**

	<i>module</i>	Module index
in	<i>pck</i>	Pointer to packet which will be send through selected module

**Returns**

Nothing

**4.13.3.8 module\_pck\_send\_receive()**

```
enum e_send_receive_status module_pck_send_receive (
    enum e_module module,
```

```
TPacket * pckin,
TPacket * pckout,
unsigned char timeout )
```

Sending and receiving packet through selected module.

#### Parameters

	<i>module</i>	Module index
in	<i>pckin</i>	Pointer to packet which will be send through selected module
out	<i>pckout</i>	Pointer to packet buffer where will be put received packet
	<i>timeout</i>	Timeout for answer waiting in milliseconds

#### Returns

Result of sending or receiveing packet

#### 4.13.3.9 module\_set\_pwr()

```
void module_set_pwr (
    enum e_module module,
    unsigned char pwr_state )
```

Switching power to each module.

#### Parameters

<i>module</i>	Module index
<i>pwr_state</i>	Power state of module. 1 - Power on, 0 - Power off

#### Returns

Nothing

#### 4.13.3.10 module\_set\_sleep()

```
void module_set_sleep (
    enum e_module module )
```

Set selected module to sleep mode.

#### Parameters

<i>module</i>	Module index
---------------	--------------



**Returns**

Nothing

**Note**

Implemented only on Semtech module and Nanotron module

**4.13.3.11 module\_sx\_present()**

```
bool module_sx_present (
    enum e_module module )
```

Semtech module detection.

**Parameters**

<i>module</i>	Module index
---------------	--------------

**Returns**

`true` if module was detected

**4.13.3.12 modules\_init()**

```
void modules_init (
    void )
```

Initialize all modules in slots.

**Returns**

Nothing

**4.14 src/modules.h File Reference**

Modules control.

```
#include <pck.h>
```

**Classes**

- struct [MODULE\\_STRUCT](#)  
*Module state in each slot.*

## Enumerations

- enum `e_module` { `MODULE_1`, `MODULE_2` }  
*Index of selected module.*
- enum `e_module_type` {  
`MODULE_OFF`, `MODULE_NOT_INIT`, `MODULE_NONE`, `MODULE_SX`,  
`MODULE_NTR`, `MODULE_DW` }  
*Module type connected in slot.*

## Functions

- void `modules_init` (void)  
*Initialize all modules in slots.*
- void `module_pck_send` (enum `e_module` module, TPacket \*pck)  
*Sending packet through selected module.*
- enum `e_send_receive_status` `module_pck_send_receive` (enum `e_module` module, TPacket \*pckin, TPacket \*pckout, unsigned char timeout)  
*Sending and receiving packet through selected module.*
- unsigned char `module_get_rssi` (enum `e_module` module)  
*Measuring Receive Signal Strength Intensity of received packet.*
- unsigned char `module_get_rssi_background` (enum `e_module` module)  
*Measuring Receive Signal Strength Intensity of background.*

## Variables

- `MODULE_STRUCT` `modules`

### 4.14.1 Detailed Description

Modules control.

Author

Ondrej Jerabek

### 4.14.2 Enumeration Type Documentation

#### 4.14.2.1 e\_module

enum `e_module`

Index of selected module.

**Enumerator**

MODULE↔ _1	Module 1 selection
MODULE↔ _2	Module 2 selection

**4.14.2.2 e\_module\_type**

```
enum e_module_type
```

Module type connected in slot.

**Enumerator**

MODULE_OFF	Module off
MODULE_NOT_INIT	Module not initialized
MODULE_NONE	No module in slot
MODULE_SX	Semtech module present
MODULE_NTR	Nanotron module present
MODULE_DW	Decawave module present

**4.14.3 Function Documentation****4.14.3.1 module\_get\_rssi()**

```
unsigned char module_get_rssi (
    enum e_module module )
```

Measuring Receive Signal Strength Intensity of received packet.

**Parameters**

<i>module</i>	Module index
---------------	--------------

**Returns**

Received signal strength

**Note**

Implemented only on Semtech module

#### 4.14.3.2 module\_get\_rssi\_background()

```
unsigned char module_get_rssi_background (
    enum e_module module )
```

Measuring Receive Signal Strength Intensity of background.

##### Parameters

<i>module</i>	Module index
---------------	--------------

##### Returns

Received signal strength

##### Note

Implemented only on Semtech module and Nanotron module

#### 4.14.3.3 module\_pck\_send()

```
void module_pck_send (
    enum e_module module,
    TPacket * pck )
```

Sending packet through selected module.

##### Parameters

	<i>module</i>	Module index
in	<i>pck</i>	Pointer to packet which will be send through selected module

##### Returns

Nothing

#### 4.14.3.4 module\_pck\_send\_receive()

```
enum e_send_receive_status module_pck_send_receive (
    enum e_module module,
    TPacket * pckin,
    TPacket * pckout,
    unsigned char timeout )
```

Sending and receiving packet through selected module.

## Parameters

	<i>module</i>	Module index
in	<i>pckin</i>	Pointer to packet which will be send through selected module
out	<i>pckout</i>	Pointer to packet buffer where will be put received packet
	<i>timeout</i>	Timeout for answer waiting in milliseconds

## Returns

Result of sending or receiveing packet

## 4.14.3.5 modules\_init()

```
void modules_init (
    void )
```

Initialize all modules in slots.

## Returns

Nothing

## 4.15 src/nanotron.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "nanotron.h"
#include "config.h"
#include <ntrxtypes.h>
#include <ntrxutil.h>
#include <ntrxranging.h>
#include "../lib/multinanotron/include/phy.h"
#include <nnsapi.h>
#include <hwclock.h>
#include "main.h"
#include "modules.h"
#include "lpc_timer.h"
#include <is_rqstat.h>
#include <type.h>
#include <25XX640.h>
#include <interrupt.h>
#include <coremem.h>
#include <uart_packet.h>
#include <ntrx_packet.h>
#include <rtcsync.h>
#include <extcomm.h>
#include <wdt.h>
```

## Macros

- `#define NTRX_SELECT(i) NTRX_####_CS(0)`
- `#define NTRX_UNSELECT(i) NTRX_####_CS(1)`

## Functions

- void **HardFault\_Handler** (void)
- void **ErrorHandler** (int16\_t err)  
*Error message handler.*
- bool **nanotron\_init** (unsigned int index)  
*Initialize one nanoLOC module.*
- bool **nanotron\_init\_all** (void)  
*Initialize all nanoLOC modules.*
- bool **nanotron\_calibrate\_all** (void)  
*Calibrate all nanoLOC modules.*
- bool **nanotron\_calibrate** (NanoGlobCTX \*ngX)  
*Calibrate one NanoLOC module.*
- void **nanotron\_rx\_enable** (NanoGlobCTX \*ngX, bool enabled)  
*Enable NanoLOC receiver.*
- void **nanotron\_broadcast\_rtc** (NanoGlobCTX \*ngX)  
*Sending RTC broadcast packet through nanoLOC module.*
- void **nanotron\_task** (NanoGlobCTX \*ngX)  
*Handling nanoLOC communication.*
- void **nanotron\_enable\_promiscuous** (unsigned int index, bool enabled)  
*Enable receiving all packets (Disable address checking)*
- **TOA\_DATA** \* **nanotron\_get\_toa** (NanoGlobCTX \*ngX)  
*Read ToA (Time of Arrival) data structure.*
- NanoGlobCTX \* **nanotron\_get\_ptr** (unsigned int index)  
*Translate nanoLOC index to pointer to NanoLOC work & configuration structure.*
- enum e\_send\_receive\_status **send\_timesync\_packet** (NanoGlobCTX \*ngX)  
*Send timesync packet for NanoLOC modules.*
- void **nanotron\_beacon** (void)  
*Sending Beacon through nanotron module.*
- bool **nanotron\_pwr\_down** (NanoGlobCTX \*ngX)  
*Set nanoLOC chip to power down mode.*
- bool **nanotron\_set\_tx\_pwr** (NanoGlobCTX \*ngX, unsigned char pwr)  
*Set nanoLOC chip transmitting power.*

## Variables

- MsgT **downMsg**  
*Message structure for data requests.*
- NanoGlobCTX **nanotrons** [NTRX\_COUNT]
- **TOA\_DATA** **toa** [NTRX\_COUNT]

### 4.15.1 Detailed Description

#### Author

Ondrej Jerabek

## 4.15.2 Function Documentation

### 4.15.2.1 ErrorHandler()

```
void ErrorHandler (
    int16_t err )
```

Error message handler.

#### Parameters

<i>err</i>	Error number
------------	--------------

#### Returns

Nothing

### 4.15.2.2 nanotron\_beacon()

```
void nanotron_beacon (
    void )
```

Sending Beacon through nanotron module.

#### Returns

Nothing

### 4.15.2.3 nanotron\_broadcast\_rtc()

```
void nanotron_broadcast_rtc (
    NanoGlobCTX * ngX )
```

Sending RTC broadcast packet through nanoLOC module.

#### Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

**Returns**

Nothing

**4.15.2.4 nanotron\_calibrate()**

```
bool nanotron_calibrate (
    NanoGlobCTX * ngX )
```

Calibrate one NanoLOC module.

**Parameters**

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

**Returns**

Calibration result. True if successful

**4.15.2.5 nanotron\_calibrate\_all()**

```
bool nanotron_calibrate_all (
    void )
```

Calibrate all nanoLOC modules.

**Returns**

Calibration result. True if successful

**4.15.2.6 nanotron\_enable\_promiscuous()**

```
void nanotron_enable_promiscuous (
    unsigned int index,
    bool enabled )
```

Enable receiving all packets (Disable address checking)

**Parameters**

<i>index</i>	NanoLOC chip index
<i>enabled</i>	Enable/disable receiving all packets



**Returns**

Nothing

**4.15.2.7 nanotron\_get\_ptr()**

```
NanoGlobCTX* nanotron_get_ptr (
    unsigned int index ) [inline]
```

Translate nanoLOC index to pointer to NanoLOC work & configuration structure.

**Parameters**

<i>index</i>	NanoLOC chip index
--------------	--------------------

**Returns**

Pointer to NanoLOC work & configuration structure

**4.15.2.8 nanotron\_get\_toa()**

```
TOA_DATA* nanotron_get_toa (
    NanoGlobCTX * ngX )
```

Read ToA (Time of Arrival) data structure.

**Parameters**

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

**Returns**

Pointer to TOA data structure

**4.15.2.9 nanotron\_init()**

```
bool nanotron_init (
    unsigned int index )
```

Initialize one nanoLOC module.

**Parameters**

<i>index</i>	NanoLOC chip index
--------------	--------------------

**Returns**

Initialization result. True if successful

**4.15.2.10 nanotron\_init\_all()**

```
bool nanotron_init_all (  
    void )
```

Initialize all nanoLOC modules.

**Returns**

Initialization result. True if successful

**4.15.2.11 nanotron\_pwr\_down()**

```
bool nanotron_pwr_down (  
    NanoGlobCTX * ngX )
```

Set nanoLOC chip to power down mode.

**Parameters**

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

**Returns**

false.

**4.15.2.12 nanotron\_rx\_enable()**

```
void nanotron_rx_enable (  
    NanoGlobCTX * ngX,  
    bool enabled )
```

Enable NanoLOC receiver.

## Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

## Returns

Nothing

## 4.15.2.13 nanotron\_set\_tx\_pwr()

```
bool nanotron_set_tx_pwr (
    NanoGlobCTX * ngX,
    unsigned char pwr )
```

Set nanoLOC chip transmitting power.

## Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
<i>pwr</i>	NanoLOC transmit power. Valid value 0 - 63

## Returns

true if set succesfully

## 4.15.2.14 nanotron\_task()

```
void nanotron_task (
    NanoGlobCTX * ngX )
```

Handling nanoLOC communication.

## Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

## Returns

Nothing

#### 4.15.2.15 send\_timesync\_packet()

```
enum e_send_receive_status send_timesync_packet (
    NanoGlobCTX * ngX )
```

Send timesync packet for NanoLOC modules.

##### Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

##### Returns

Result of packet sending

##### Warning

Not implemented yet. For testing purpose, sends only device status

## 4.16 src/nanotron.h File Reference

Functions for controlling NanoLOC.

```
#include <NanoGlobal.h>
```

### Classes

- struct [TOA\\_DATA](#)  
*ToA (Time of Arrival) structure.*

### Macros

- #define [SendMsg](#) PDSap  
*Data request to the next lower layer.*

### Functions

- bool [nanotron\\_calibrate\\_all](#) (void)  
*Calibrate all nanoLOC modules.*
- bool [nanotron\\_init\\_all](#) (void)  
*Initialize all nanoLOC modules.*
- bool [nanotron\\_init](#) (unsigned int index)  
*Initialize one nanoLOC module.*
- void [nanotron\\_task](#) (NanoGlobCTX \*ngX)  
*Handling nanoLOC communication.*
- bool [nanotron\\_calibrate](#) (NanoGlobCTX \*ngX)

- *Calibrate one NanoLOC module.*
- void [nanotron\\_rx\\_enable](#) (NanoGlobCTX \*ngX, bool enabled)
- *Enable NanoLOC receiver.*
- void [nanotron\\_broadcast\\_rtc](#) (NanoGlobCTX \*ngX)
- *Sending RTC broadcast packet through nanoLOC module.*
- void [nanotron\\_enable\\_promiscuous](#) (unsigned int index, bool enabled)
- *Enable receiving all packets (Disable address checking)*
- [TOA\\_DATA](#) \* [nanotron\\_get\\_toa](#) (NanoGlobCTX \*ngX)
- *Read ToA (Time of Arrival) data structure.*
- NanoGlobCTX \* [nanotron\\_get\\_ptr](#) (unsigned int index)
- *Translate nanoLOC index to pointer to NanoLOC work & configuration structure.*
- bool [nanotron\\_set\\_tx\\_pwr](#) (NanoGlobCTX \*ngX, unsigned char pwr)
- *Set nanoLOC chip transmitting power.*
- void [nanotron\\_beacon](#) (void)
- *Sending Beacon through nanotron module.*

## Variables

- MsgT [downMsg](#)
- *Message structure for data requests.*
- [TOA\\_DATA](#) [toa](#) [NTRX\_COUNT]

### 4.16.1 Detailed Description

Functions for controlling NanoLOC.

#### Author

Ondrej Jerabek

### 4.16.2 Macro Definition Documentation

#### 4.16.2.1 SendMsg

```
#define SendMsg PDSap
```

Data request to the next lower layer.

#### Note

This function calls the next lower layer to request a data transmission.

### 4.16.3 Function Documentation

#### 4.16.3.1 nanotron\_beacon()

```
void nanotron_beacon (
    void )
```

Sending Beacon through nanotron module.

##### Returns

Nothing

#### 4.16.3.2 nanotron\_broadcast\_rtc()

```
void nanotron_broadcast_rtc (
    NanoGlobCTX * ngX )
```

Sending RTC broadcast packet through nanoLOC module.

##### Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

##### Returns

Nothing

#### 4.16.3.3 nanotron\_calibrate()

```
bool nanotron_calibrate (
    NanoGlobCTX * ngX )
```

Calibrate one NanoLOC module.

##### Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

##### Returns

Calibration result. True if successful

#### 4.16.3.4 nanotron\_calibrate\_all()

```
bool nanotron_calibrate_all (
    void )
```

Calibrate all nanoLOC modules.

##### Returns

Calibration result. True if successful

#### 4.16.3.5 nanotron\_enable\_promiscuous()

```
void nanotron_enable_promiscuous (
    unsigned int index,
    bool enabled )
```

Enable receiving all packets (Disable address checking)

##### Parameters

<i>index</i>	NanoLOC chip index
<i>enabled</i>	Enable/disable receiving all packets

##### Returns

Nothing

#### 4.16.3.6 nanotron\_get\_ptr()

```
NanoGlobCTX* nanotron_get_ptr (
    unsigned int index ) [inline]
```

Translate nanoLOC index to pointer to NanoLOC work & configuration structure.

##### Parameters

<i>index</i>	NanoLOC chip index
--------------	--------------------

##### Returns

Pointer to NanoLOC work & configuration structure

#### 4.16.3.7 nanotron\_get\_toa()

```
TOA_DATA* nanotron_get_toa (
    NanoGlobCTX * ngX )
```

Read ToA (Time of Arrival) data structure.

##### Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

##### Returns

Pointer to TOA data structure

#### 4.16.3.8 nanotron\_init()

```
bool nanotron_init (
    unsigned int index )
```

Initialize one nanoLOC module.

##### Parameters

<i>index</i>	NanoLOC chip index
--------------	--------------------

##### Returns

Initialization result. True if successful

#### 4.16.3.9 nanotron\_init\_all()

```
bool nanotron_init_all (
    void )
```

Initialize all nanoLOC modules.

##### Returns

Initialization result. True if successful

#### 4.16.3.10 nanotron\_rx\_enable()

```
void nanotron_rx_enable (
    NanoGlobCTX * ngX,
    bool enabled )
```

Enable NanoLOC receiver.



## Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

## Returns

Nothing

## 4.16.3.11 nanotron\_set\_tx\_pwr()

```
bool nanotron_set_tx_pwr (
    NanoGlobCTX * ngX,
    unsigned char pwr )
```

Set nanoLOC chip transmitting power.

## Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
<i>pwr</i>	NanoLOC transmit power. Valid value 0 - 63

## Returns

true if set succesfully

## 4.16.3.12 nanotron\_task()

```
void nanotron_task (
    NanoGlobCTX * ngX )
```

Handling nanoLOC communication.

## Parameters

<i>ngX</i>	Pointer to NanoLOC work & configuration structure
------------	---

## Returns

Nothing

## 4.17 src/pindef.c File Reference

```
#include "config.h"
#include <chip.h>
```

```
#include "iocon_17xx_40xx.h"
#include "gpioint_17xx_40xx.h"
#include "spi_wrapper.h"
#include "uart_wrapper.h"
#include "main.h"
```

## Functions

- void [pindef\\_init](#) (void)  
*Initialize default pin state and mode.*

## Variables

- STATIC const PINMUX\_GRP\_T [pinmuxing](#) []  
*Default pin modes.*

### 4.17.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.17.2 Function Documentation

#### 4.17.2.1 [pindef\\_init\(\)](#)

```
void pindef_init (  
    void )
```

Initialize default pin state and mode.

#### Returns

Nothing

## 4.18 src/pindef.h File Reference

Pin definitions and configuration.

```
#include <type.h>
#include <delay.h>
```

## Macros

- `#define LED_SYS_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 1, 30, val)`
- `#define LED_SYS(val) Chip_GPIO_WritePortBit(LPC_GPIO, 1, 30, val)`
- `#define LED_STAT_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 3, 23, val)`
- `#define LED_STAT(val) Chip_GPIO_WritePortBit(LPC_GPIO, 3, 23, val)`
- `#define LEDG_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 1, 21, val)`
- `#define LEDG(val) Chip_GPIO_WritePortBit(LPC_GPIO, 1, 21, val)`
- `#define LEDR_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 1, 20, val)`
- `#define LEDR(val) Chip_GPIO_WritePortBit(LPC_GPIO, 1, 20, val)`
- `#define LEDY2_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 1, 18, val)`
- `#define LEDY2(val) Chip_GPIO_WritePortBit(LPC_GPIO, 1, 18, val)`
- `#define LEDY1_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 1, 19, val)`
- `#define LEDY1(val) Chip_GPIO_WritePortBit(LPC_GPIO, 1, 19, val)`
- `#define BUZZ_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 3, val)`
- `#define BUZZ(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 3, val)`
- `#define CHRG_EN1_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 5, 4, val)`
- `#define CHRG_EN1(val) Chip_GPIO_WritePortBit(LPC_GPIO, 5, 4, val)`
- `#define CHRG_EN2_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 5, 0, val)`
- `#define CHRG_EN2(val) Chip_GPIO_WritePortBit(LPC_GPIO, 5, 0, val)`
- `#define CHRG_CE_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 1, 31, val)`
- `#define CHRG_CE(val) Chip_GPIO_WritePortBit(LPC_GPIO, 1, 31, val)`
- `#define CHRG_PWR_GOOD Chip_GPIO_GetPinState(LPC_GPIO, 4, 29)`
- `#define CHRG_CHARGING Chip_GPIO_GetPinState(LPC_GPIO, 5, 1)`
- `#define BUTTON_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 2, val)`
- `#define BUTTON Chip_GPIO_GetPinState(LPC_GPIO, 0, 2)`
- `#define POWER_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 28, val)`
- `#define POWER(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 28, !val)`
- `#define DS_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 1, 22, val)`
- `#define DS_OUT(val) Chip_GPIO_WritePortBit(LPC_GPIO, 1, 22, val)`
- `#define DS_IN Chip_GPIO_GetPinState(LPC_GPIO, 1, 22)`
- `#define SPI_EEPROM_CS_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 16, val)`
- `#define SPI_EEPROM_CS Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 16, 0)`
- `#define spi_eeprom_select() Chip_GPIO_WritePortBit(LPC_GPIO, 0, 16, 0)`
- `#define spi_eeprom_unselect() Chip_GPIO_WritePortBit(LPC_GPIO, 0, 16, 1)`
- `#define EXTFLASH_CS_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 5, 2, val)`
- `#define EXTFLASH_CS_IN Chip_GPIO_WriteDirBit(LPC_GPIO, 5, 2, 0)`
- `#define EXTFLASH_CS(val) Chip_GPIO_WritePortBit(LPC_GPIO, 5, 2, val)`
- `#define EXTFLASH_SELECT(addr) EXTFLASH_CS(0), delayus_fcn(10)`
- `#define EXTFLASH_UNSELECT() EXTFLASH_CS(1), delayus_fcn(10)`
- `#define SD_DETECT_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 1, val)`
- `#define SD_DETECT Chip_GPIO_GetPinState(LPC_GPIO, 0, 1)`
- `#define USB_CON_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 14, val)`
- `#define USB_CON(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 14, val)`
- `#define MODULE1_RESET_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 9, val)`
- `#define MODULE1_RESET(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 9, val)`
- `#define MODULE2_RESET_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 3, val)`
- `#define MODULE2_RESET(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 3, val)`
- `#define MODULE1_PWR_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 5, 3, val)`
- `#define MODULE1_PWR(val) Chip_GPIO_WritePortBit(LPC_GPIO, 5, 3, val)`
- `#define MODULE2_PWR_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 4, 28, val)`
- `#define MODULE2_PWR(val) Chip_GPIO_WritePortBit(LPC_GPIO, 4, 28, val)`
- `#define IRQ_01_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 4, val)`
- `#define IRQ_11_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 5, val)`
- `#define IRQ01_PIN 4`

- #define **IRQ11\_PIN** 5
- #define **IRQ\_01** Chip\_GPIO\_GetPinState(LPC\_GPIO, 2, 4)
- #define **IRQ\_11** Chip\_GPIO\_GetPinState(LPC\_GPIO, 2, 5)
- #define **SX\_CONFIG\_SELECT1\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 7, val)
- #define **SX\_CONFIG\_SELECT1**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 2, 7, 0), Chip\_GPIO\_WritePort↵  
Bit(LPC\_GPIO, 0, 6, 0)
- #define **SX\_CONFIG\_UNSELECT1**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 2, 7, 1), Chip\_GPIO\_Write↵  
PortBit(LPC\_GPIO, 0, 6, 1)
- #define **SX\_DATA\_SELECT1\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 0, 6, val)
- #define **SX\_DATA\_SELECT1**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 6, 0)
- #define **SX\_DATA\_UNSELECT1**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 6, 1)
- #define **NSS\_CONFIG1** Chip\_GPIO\_ReadPortBit(LPC\_GPIO, 2, 7)
- #define **NSS\_DATA1** Chip\_GPIO\_ReadPortBit(LPC\_GPIO, 0, 6)
- #define **SX\_PLLOCK1\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 3, 25, val)
- #define **SX\_PLLOCK1** Chip\_GPIO\_GetPinState(LPC\_GPIO, 3, 25)
- #define **SX\_RESET1\_DIR** MODULE0\_RESET\_DIR
- #define **SX\_RESET1**(val) MODULE0\_RESET(val)
- #define **SX\_EE\_CS1\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 6, val)
- #define **SX\_EE\_CS1**(val) Chip\_GPIO\_WritePortBit(LPC\_GPIO, 2, 6, val)
- #define **SX\_EE\_CS1\_PIN** Chip\_GPIO\_ReadPortBit(LPC\_GPIO, 2, 6)
- #define **IRQ\_02\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 0, val)
- #define **IRQ\_12\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 1, val)
- #define **IRQ02\_PIN** 0
- #define **IRQ12\_PIN** 1
- #define **IRQ\_02** Chip\_GPIO\_GetPinState(LPC\_GPIO, 2, 0)
- #define **IRQ\_12** Chip\_GPIO\_GetPinState(LPC\_GPIO, 2, 1)
- #define **SX\_CONFIG\_SELECT2\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 0, 4, val)
- #define **SX\_CONFIG\_SELECT2**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 4, 0), Chip\_GPIO\_WritePort↵  
Bit(LPC\_GPIO, 0, 5, 0)
- #define **SX\_CONFIG\_UNSELECT2**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 4, 1), Chip\_GPIO\_Write↵  
PortBit(LPC\_GPIO, 0, 5, 1)
- #define **SX\_DATA\_SELECT2\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 0, 5, val)
- #define **SX\_DATA\_SELECT2**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 5, 0)
- #define **SX\_DATA\_UNSELECT2**() Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 5, 1)
- #define **NSS\_CONFIG2** Chip\_GPIO\_ReadPortBit(LPC\_GPIO, 0, 4)
- #define **NSS\_DATA2** Chip\_GPIO\_ReadPortBit(LPC\_GPIO, 0, 5)
- #define **SX\_PLLOCK2\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 3, 26, val)
- #define **SX\_PLLOCK2** Chip\_GPIO\_GetPinState(LPC\_GPIO, 3, 26)
- #define **SX\_RESET2\_DIR** MODULE1\_RESET\_DIR
- #define **SX\_RESET2**(val) MODULE1\_RESET(val)
- #define **SX\_EE\_CS2\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 2, val)
- #define **SX\_EE\_CS2**(val) Chip\_GPIO\_WritePortBit(LPC\_GPIO, 2, 2, val)
- #define **SX\_EE\_CS2\_PIN** Chip\_GPIO\_ReadPortBit(LPC\_GPIO, 2, 2)
- #define **IRQ\_DW0\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 4, val)
- #define **IRQ\_DW0** Chip\_GPIO\_GetPinState(LPC\_GPIO, 2, 4)
- #define **DW\_D3\_0\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 5, val)
- #define **DW\_D3\_0**(val) Chip\_GPIO\_WritePortBit(LPC\_GPIO, 2, 5, val)
- #define **DW\_HGM\_0\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 7, val)
- #define **DW\_HGM\_0**(val) Chip\_GPIO\_WritePortBit(LPC\_GPIO, 2, 7, val)
- #define **DW\_EN\_0\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 6, val)
- #define **DW\_EN\_0**(val) Chip\_GPIO\_WritePortBit(LPC\_GPIO, 2, 6, val)
- #define **DW\_CS\_0\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 0, 6, val)
- #define **DW\_CS\_0**(val) Chip\_GPIO\_WritePortBit(LPC\_GPIO, 0, 6, val)
- #define **IRQ\_DW1\_DIR**(val) Chip\_GPIO\_WriteDirBit(LPC\_GPIO, 2, 0, val)
- #define **IRQ\_DW1** Chip\_GPIO\_GetPinState(LPC\_GPIO, 2, 0)

- `#define DW_D3_1_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 1, val)`
- `#define DW_D3_1(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 1, val)`
- `#define DW_HGM_1_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 4, val)`
- `#define DW_HGM_1(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 4, val)`
- `#define DW_EN_1_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 2, val)`
- `#define DW_EN_1(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 2, val)`
- `#define DW_CS_1_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 5, val)`
- `#define DW_CS_1(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 5, val)`
- `#define DW_WSYNC_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 8, val)`
- `#define DW_WSYNC(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 8, val)`
- `#define NTRX_1_N_RESET_DIR(val) MODULE1_RESET_DIR(val)`
- `#define NTRX_1_N_RESET(val) MODULE1_RESET(val)`
- `#define NTRX_1_CS_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 6, val)`
- `#define NTRX_1_CS(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 6, val)`
- `#define NTRX_1_PAEN_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 6, val)`
- `#define NTRX_1_PAEN(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 6, val)`
- `#define NTRX_1_EN_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 6, val)`
- `#define NTRX_1_EN(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 6, val)`
- `#define NTRX_1_HGM_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 7, val)`
- `#define NTRX_1_HGM(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 7, val)`
- `#define NTRX_1_IRQ_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 4, val)`
- `#define NTRX_1_IRQ Chip_GPIO_GetPinState(LPC_GPIO, 2, 4)`
- `#define NTRX_1_VCC_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 3, 25, val)`
- `#define NTRX_1_VCC Chip_GPIO_GetPinState(LPC_GPIO, 3, 25)`
- `#define NTRX_1_M_RESET`
- `#define NTRX_1_D3_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 5, val)`
- `#define NTRX_1_D3 Chip_GPIO_GetPinState(LPC_GPIO, 2, 5)`
- `#define NTRX_2_N_RESET_DIR(val) MODULE2_RESET_DIR(val)`
- `#define NTRX_2_N_RESET(val) MODULE2_RESET(val)`
- `#define NTRX_2_CS_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 5, val)`
- `#define NTRX_2_CS(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 5, val)`
- `#define NTRX_2_PAEN_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 2, val)`
- `#define NTRX_2_PAEN(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 2, val)`
- `#define NTRX_2_EN_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 2, val)`
- `#define NTRX_2_EN(val) Chip_GPIO_WritePortBit(LPC_GPIO, 2, 2, val)`
- `#define NTRX_2_HGM_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 0, 4, val)`
- `#define NTRX_2_HGM(val) Chip_GPIO_WritePortBit(LPC_GPIO, 0, 4, val)`
- `#define NTRX_2_IRQ_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 0, val)`
- `#define NTRX_2_IRQ Chip_GPIO_GetPinState(LPC_GPIO, 2, 0)`
- `#define NTRX_2_VCC_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 3, 26, val)`
- `#define NTRX_2_VCC Chip_GPIO_GetPinState(LPC_GPIO, 3, 26)`
- `#define NTRX_2_M_RESET`
- `#define NTRX_2_D3_DIR(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 2, 1, val)`
- `#define NTRX_2_D3 Chip_GPIO_GetPinState(LPC_GPIO, 2, 1)`
- `#define INIT_IO()`

*Initialization of I/O pins and set to default values.*

## Functions

- void `pindef_init` (void)

*Initialize default pin state and mode.*

### 4.18.1 Detailed Description

Pin definitions and configuration.

#### Author

Ondrej Jerabek

### 4.18.2 Function Documentation

#### 4.18.2.1 pindef\_init()

```
void pindef_init (  
    void )
```

Initialize default pin state and mode.

#### Returns

Nothing

## 4.19 src/power\_ctrl.c File Reference

```
#include "power_ctrl.h"  
#include "pindef.h"  
#include "main.h"  
#include <adc_17xx_40xx.h>  
#include <wdt.h>
```

### Classes

- struct [AvgStruct](#)  
*Averaging structure.*

### Functions

- void [avg\\_handle](#) ([AvgStruct](#) \*avg\_t, unsigned short in\_val, unsigned short \*result)  
*Averaging measured values.*
- void [set\\_chrg\\_mode](#) (enum [e\\_chrg\\_mode](#) mode)  
*Set charger power mode.*
- void [pwr\\_suicide](#) (void)  
*Device power off.*
- void [pwr\\_meas\\_vals](#) (void)  
*Handling ADC measurement.*
- void [pwr\\_indication](#) (void)  
*Indication power states on LED and Buzzer.*
- void [pwr\\_handle](#) (void)  
*Handling power control and state indication.*

## Variables

- [AvgStruct](#) `avg` [4]

### 4.19.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.19.2 Function Documentation

#### 4.19.2.1 `avg_handle()`

```
void avg_handle (
    AvgStruct * avg_t,
    unsigned short in_val,
    unsigned short * result )
```

Averaging measured values.

#### Parameters

<i>avg_t</i>	Pointer to averaging structure
<i>in_val</i>	Input value for averaging
<i>result</i>	Pointer to variable where result be stored

#### Returns

Nothing

#### 4.19.2.2 `pwr_handle()`

```
void pwr_handle (
    void )
```

Handling power control and state indication.

#### Returns

Nothing

#### 4.19.2.3 pwr\_indication()

```
void pwr_indication (  
    void )
```

Indication power states on LED and Buzzer.

##### Returns

Nothing

#### 4.19.2.4 pwr\_meas\_vals()

```
void pwr_meas_vals (  
    void )
```

Handling ADC measurement.

##### Returns

Nothing

#### 4.19.2.5 pwr\_suicide()

```
void pwr_suicide (  
    void )
```

Device power off.

##### Returns

Nothing

##### Note

If device powered from external power, does nothing

#### 4.19.2.6 set\_chrg\_mode()

```
void set_chrg_mode (  
    enum e_chrg_mode mode )
```

Set charger power mode.



## Parameters

<i>mode</i>	Current mode. Valid values available in enum <code>e_chrg_mode</code>
-------------	---

## Returns

Nothing

## 4.20 src/power\_ctrl.h File Reference

Power and charging control.

## Enumerations

- enum `e_chrg_mode` {  
    `CHRG_OFF`, `CHRG_100MA`, `CHRG_500MA`, `CHRG_600MA`,  
    `CHRG_BATT_ONLY` }  
    Charger configuration.

## Functions

- void `set_chrg_mode` (enum `e_chrg_mode` mode)  
    Set charger power mode.
- void `pwr_meas_vals` (void)  
    Handling ADC measurement.
- void `pwr_suicide` (void)  
    Device power off.
- void `pwr_handle` (void)  
    Handling power control and state indication.

### 4.20.1 Detailed Description

Power and charging control.

## Author

Ondrej Jerabek

### 4.20.2 Enumeration Type Documentation

#### 4.20.2.1 e\_chrg\_mode

enum `e_chrg_mode`

Charger configuration.

**Enumerator**

CHRG_OFF	Charging off
CHRG_100MA	Charging on, maximum input current 100mA
CHRG_500MA	Charging on, maximum input current 500mA
CHRG_600MA	Charging on, maximum input current 600mA
CHRG_BATT_ONLY	Charging off, Device powered from battery only

**4.20.3 Function Documentation****4.20.3.1 pwr\_handle()**

```
void pwr_handle (  
    void )
```

Handling power control and state indication.

**Returns**

Nothing

**4.20.3.2 pwr\_meas\_vals()**

```
void pwr_meas_vals (  
    void )
```

Handling ADC measurement.

**Returns**

Nothing

**4.20.3.3 pwr\_suicide()**

```
void pwr_suicide (  
    void )
```

Device power off.

**Returns**

Nothing

**Note**

If device powered from external power, does nothing

## 4.20.3.4 set\_chrg\_mode()

```
void set_chrg_mode (
    enum e_chrg_mode mode )
```

Set charger power mode.

## Parameters

<i>mode</i>	Current mode. Valid values available in enum e_chrg_mode
-------------	--

## Returns

Nothing

## 4.21 src/routing.c File Reference

```
#include "routing.h"
```

## Functions

- bool [routing\\_table\\_update](#) (unsigned int addr, enum e\_packet\_source\_device interface)  
*Updating item in routing table.*
- void [routing\\_table\\_timeout](#) ()  
*Handle timers of the items in table.*
- enum e\_packet\_source\_device [routing\\_table\\_find](#) (unsigned int addr)  
*Finding item in routing table.*
- void [routing\\_table\\_device\\_del](#) (unsigned int addr)  
*Delete item from routing table.*

## Variables

- [ROUTING\\_TABLE\\_ITEM](#) **routing\_table** [ROUTING\_TABLE\_SIZE]

## 4.21.1 Detailed Description

## Author

Ondrej Jerabek

## Note

Not fully implemented yet

## 4.21.2 Function Documentation

### 4.21.2.1 `routing_table_device_del()`

```
void routing_table_device_del (  
    unsigned int addr )
```

Delete item from routing table.

**Parameters**

<i>addr</i>	Device address to delete
-------------	--------------------------

**Returns**

Nothing

**4.21.2.2 routing\_table\_find()**

```
enum e_packet_source_device routing_table_find (  
    unsigned int addr )
```

Finding item in routing table.

**Parameters**

<i>addr</i>	Device address to find
-------------	------------------------

**Returns**

Source interface

**4.21.2.3 routing\_table\_timeout()**

```
void routing_table_timeout (  
    void )
```

Handle timers of the items in table.

**Returns**

Nothing

**Note**

Should be called each second

**4.21.2.4 routing\_table\_update()**

```
bool routing_table_update (  
    unsigned int addr,  
    enum e_packet_source_device interface )
```

Updating item in routing table.

#### Parameters

<i>addr</i>	Sorce device address
<i>interface</i>	Interface where is the device present

#### Returns

Status of item update. True if item successfully updated/added

## 4.22 src/routing.h File Reference

Routing function.

```
#include "config.h"  
#include <packet_handle.h>  
#include <type.h>
```

#### Classes

- struct [ROUTING\\_TABLE\\_ITEM](#)  
*Routing table item structure.*

#### Macros

- #define **ROUTING\_TABLE\_SIZE** 32
- #define **ROUTING\_TABLE\_TIMEOUT** 60

#### Functions

- bool [routing\\_table\\_update](#) (unsigned int addr, enum e\_packet\_source\_device interface)  
*Updating item in routing table.*
- void [routing\\_table\\_timeout](#) (void)  
*Handle timers of the items in table.*
- enum e\_packet\_source\_device [routing\\_table\\_find](#) (unsigned int addr)  
*Finding item in routing table.*
- void [routing\\_table\\_device\\_del](#) (unsigned int addr)  
*Delete item from routing table.*

#### Variables

- [ROUTING\\_TABLE\\_ITEM](#) **routing\_table** [ROUTING\_TABLE\_SIZE]

### 4.22.1 Detailed Description

Routing function.

#### Author

Ondrej Jerabek

#### Note

Not fully implemented yet

### 4.22.2 Function Documentation

#### 4.22.2.1 routing\_table\_device\_del()

```
void routing_table_device_del (  
    unsigned int addr )
```

Delete item from routing table.

#### Parameters

<i>addr</i>	Device address to delete
-------------	--------------------------

#### Returns

Nothing

#### 4.22.2.2 routing\_table\_find()

```
enum e_packet_source_device routing_table_find (  
    unsigned int addr )
```

Finding item in routing table.

#### Parameters

<i>addr</i>	Device address to find
-------------	------------------------

#### Returns

Source interface

#### 4.22.2.3 routing\_table\_timeout()

```
void routing_table_timeout (
    void )
```

Handle timers of the items in table.

##### Returns

Nothing

##### Note

Should be called each second

#### 4.22.2.4 routing\_table\_update()

```
bool routing_table_update (
    unsigned int addr,
    enum e_packet_source_device interface )
```

Updating item in routing table.

##### Parameters

<i>addr</i>	Sorce device address
<i>interface</i>	Interface where is the device present

##### Returns

Status of item update. True if item successfully updated/added

## 4.23 src/rssi.c File Reference

```
#include "rssi.h"
#include "config.h"
#include "main.h"
#include <sx_rssi.h>
```

### Functions

- void [rssi\\_table\\_tick](#) (void)



*Handle timers of the items in table.*

- void `rssi_table_add` (unsigned int *addr*, unsigned char *module*, unsigned char *rssi*)

*Adding item to RSSI table.*

- unsigned char `rssi_table_get` (unsigned char \**buffer*, unsigned char *index*, unsigned char *max\_rec\_cnt*)

*Put RSSI table to buffer (Prepare it for send)*

## Variables

- `rssi_table_t rssi_table` [RSSI\_TABLE\_SIZE]

### 4.23.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.23.2 Function Documentation

#### 4.23.2.1 `rssi_table_add()`

```
void rssi_table_add (
    unsigned int addr,
    unsigned char module,
    unsigned char rssi )
```

Adding item to RSSI table.

#### Parameters

<i>addr</i>	Device address
<i>module</i>	Module which received packet
<i>rssi</i>	Received packet RSSI

#### Returns

Nothing

#### 4.23.2.2 `rssi_table_get()`

```
unsigned char rssi_table_get (
    unsigned char * buffer,
    unsigned char index,
    unsigned char max_rec_cnt )
```

Put RSSI table to buffer (Prepare it for send)

**Parameters**

<i>buffer</i>	Pointer to buffer where RSSI table will be copied
<i>index</i>	Index in RSSI table where the copying starts
<i>max_rec_cnt</i>	Maximum number of items copied to buffer

**Returns**

Occupied size in bytes

**4.23.2.3 rssi\_table\_tick()**

```
void rssi_table_tick (
    void )
```

Handle timers of the items in table.

**Returns**

Nothing

**Note**

Should be called each second

**4.24 src/rssi.h File Reference**

Receive Signal Strength Intensity Table.

```
#include <type.h>
#include "config.h"
```

**Classes**

- struct [rssi\\_table\\_t](#)  
*RSSI table item structure.*

**Macros**

- `#define RSSI_NOT_SUPPORTED 255`

## Functions

- void [rssi\\_table\\_tick](#) (void)  
*Handle timers of the items in table.*
- void [rssi\\_table\\_add](#) (unsigned int addr, unsigned char module, unsigned char rssi)  
*Adding item to RSSI table.*
- unsigned char [rssi\\_table\\_get](#) (unsigned char \*buffer, unsigned char index, unsigned char max\_rec\_cnt)  
*Put RSSI table to buffer (Prepare it for send)*

## Variables

- [rssi\\_table\\_t](#) **rssi\_table** [RSSI\_TABLE\_SIZE]

### 4.24.1 Detailed Description

Receive Signal Strength Intensity Table.

#### Author

Ondrej Jerabek

### 4.24.2 Function Documentation

#### 4.24.2.1 [rssi\\_table\\_add\(\)](#)

```
void rssi_table_add (  
    unsigned int addr,  
    unsigned char module,  
    unsigned char rssi )
```

Adding item to RSSI table.

#### Parameters

<i>addr</i>	Device address
<i>module</i>	Module which received packet
<i>rssi</i>	Received packet RSSI

#### Returns

Nothing

#### 4.24.2.2 rssi\_table\_get()

```
unsigned char rssi_table_get (
    unsigned char * buffer,
    unsigned char index,
    unsigned char max_rec_cnt )
```

Put RSSI table to buffer (Prepare it for send)

##### Parameters

<i>buffer</i>	Pointer to buffer where RSSI table will be copied
<i>index</i>	Index in RSSI table where the copying starts
<i>max_rec_cnt</i>	Maximum number of items copied to buffer

##### Returns

Occupied size in bytes

#### 4.24.2.3 rssi\_table\_tick()

```
void rssi_table_tick (
    void )
```

Handle timers of the items in table.

##### Returns

Nothing

##### Note

Should be called each second

## 4.25 src/spi\_wrapper.c File Reference

```
#include <stdint.h>
#include "config.h"
#include <chip.h>
#include "spi_wrapper.h"
#include "ssp_17xx_40xx.h"
```

## Functions

- void [SPI\\_Master\\_Init](#) (unsigned char index, unsigned int freq)  
*Initialize SPI in Master mode.*
- void [SPI\\_Wait\\_Busy](#) (unsigned char index)  
*Waiting for complete SPI transfer.*
- void [SPI\\_Write](#) (unsigned char index, unsigned short data)  
*Send data through SPI.*
- unsigned short [SPI\\_Read\\_Write](#) (unsigned char index, unsigned short data)  
*Send and receive data through SPI.*

### 4.25.1 Detailed Description

#### Author

Ondrej Jerabek

### 4.25.2 Function Documentation

#### 4.25.2.1 SPI\_Master\_Init()

```
void SPI_Master_Init (
    unsigned char index,
    unsigned int freq )
```

Initialize SPI in Master mode.

#### Parameters

<i>index</i>	Index of SPI peripheral
<i>freq</i>	Frequency of SPI clk

#### Returns

Nothing

#### 4.25.2.2 SPI\_Read\_Write()

```
unsigned short SPI_Read_Write (
    unsigned char index,
    unsigned short data )
```

Send and receive data through SPI.

**Parameters**

<i>index</i>	Index of SPI peripheral
<i>data</i>	Data to send

**Returns**

Received data

**4.25.2.3 SPI\_Wait\_Busy()**

```
void SPI_Wait_Busy (
    unsigned char index )
```

Waiting for complete SPI transfer.

**Parameters**

<i>index</i>	Index of SPI peripheral
--------------	-------------------------

**Returns**

Nothing

**4.25.2.4 SPI\_Write()**

```
void SPI_Write (
    unsigned char index,
    unsigned short data )
```

Send data through SPI.

**Parameters**

<i>index</i>	Index of SPI peripheral
<i>data</i>	Data to send

**Returns**

Nothing

**4.26 src/spi\_wrapper.h File Reference**

Wrapper for using more SPIs.

## Macros

- `#define __SPI(index, name) SPI_##index##_##name`
- `#define _SPI(index, name) __SPI(index, name)`
- `#define SPI_FREQ _SPI(SPI_INDEX, FREQ)`
- `#define spi_init() SPI_Master_Init(SPI_INDEX, SPI_FREQ)`
- `#define spi_write(data) SPI_Write(SPI_INDEX, data)`
- `#define spi_read_write(data) SPI_Read_Write(SPI_INDEX, data)`
- `#define spi_read() SPI_Read_Write(SPI_INDEX, 0x00)`
- `#define spi_wait_busy() SPI_Wait_Busy(SPI_INDEX)`

## Functions

- void `SPI_Master_Init` (unsigned char index, unsigned int freq)  
*Initialize SPI in Master mode.*
- void `SPI_Write` (unsigned char index, unsigned short data)  
*Send data through SPI.*
- unsigned short `SPI_Read_Write` (unsigned char index, unsigned short data)  
*Send and receive data through SPI.*
- void `SPI_Wait_Busy` (unsigned char index)  
*Waiting for complete SPI transfer.*

### 4.26.1 Detailed Description

Wrapper for using more SPIs.

#### Author

Ondrej Jerabek

### 4.26.2 Function Documentation

#### 4.26.2.1 SPI\_Master\_Init()

```
void SPI_Master_Init (
    unsigned char index,
    unsigned int freq )
```

Initialize SPI in Master mode.

#### Parameters

<i>index</i>	Index of SPI peripheral
<i>freq</i>	Frequency of SPI clk

**Returns**

Nothing

**4.26.2.2 SPI\_Read\_Write()**

```
unsigned short SPI_Read_Write (
    unsigned char index,
    unsigned short data )
```

Send and receive data through SPI.

**Parameters**

<i>index</i>	Index of SPI peripheral
<i>data</i>	Data to send

**Returns**

Read data

**Parameters**

<i>index</i>	Index of SPI peripheral
<i>data</i>	Data to send

**Returns**

Received data

**4.26.2.3 SPI\_Wait\_Busy()**

```
void SPI_Wait_Busy (
    unsigned char index )
```

Waiting for complete SPI transfer.

**Parameters**

<i>index</i>	Index of SPI peripheral
--------------	-------------------------

**Returns**

Nothing



#### 4.26.2.4 SPI\_Write()

```
void SPI_Write (
    unsigned char index,
    unsigned short data )
```

Send data through SPI.

##### Parameters

<i>index</i>	Index of SPI peripheral
<i>data</i>	Data to send

##### Returns

Nothing



# Index

- ADC\_init
  - main.c, [64](#)
- ARM\_Ethernet\_Reader.h
  - e\_cfg\_type, [30](#)
  - e\_st\_type, [31](#)
- addr
  - CommTestSettStruct, [9](#)
  - ROUTING\_TABLE\_ITEM, [24](#)
  - rss\_i\_table\_t, [25](#)
- addr\_dest
  - CommStruct, [6](#)
- answer\_handle
  - extcomm.c, [51](#)
- answer\_id
  - CommStruct, [6](#)
- answer\_ms\_tick
  - extcomm.c, [51](#)
  - extcomm.h, [58](#)
- answer\_timeout\_flag
  - CommStruct, [6](#)
- answer\_timeout\_handle
  - extcomm.c, [52](#)
  - extcomm.h, [59](#)
- answer\_tmr
  - CommStruct, [6](#)
- avg\_handle
  - power\_ctrl.c, [103](#)
- AvgStruct, [5](#)
  - samples, [5](#)
  - sum, [5](#)
- awaiting\_answer
  - CommStruct, [7](#)
- batt\_voltage
  - EthernetReader\_State, [14](#)
- beacon\_broadcast
  - extcomm.c, [52](#)
  - extcomm.h, [59](#)
- bootloader
  - Timer\_struct, [26](#)
- button
  - Timer\_struct, [26](#)
- buzz
  - Timer\_struct, [26](#)
- cfg.c
  - change\_channel, [32](#)
  - device\_load\_cfg\_defaults, [32](#)
  - device\_load\_configuration, [32](#)
  - dw\_load\_cfg\_defaults, [33](#)
  - dw\_load\_configuration, [33](#)
  - fix\_version, [33](#)
  - load\_configuration, [34](#)
  - ntr\_load\_cfg\_defaults, [34](#)
  - ntr\_load\_configuration, [35](#)
  - save\_configuration, [35](#)
  - sx\_load\_cfg\_defaults, [35](#)
  - sx\_load\_configuration, [36](#)
  - update\_configuration, [36](#)
- cfg.h
  - fix\_version, [37](#)
  - load\_configuration, [37](#)
  - save\_configuration, [39](#)
  - update\_configuration, [39](#)
- change\_channel
  - cfg.c, [32](#)
- charging
  - EthernetReader\_State, [14](#)
- Chip\_USB\_Init
  - main.c, [64](#)
  - main.h, [71](#)
- chrg\_current
  - EthernetReader\_State, [14](#)
- chrg\_mode
  - EthernetReader\_State, [14](#)
- clear\_answer\_timer
  - extcomm.c, [52](#)
  - extcomm.h, [59](#)
- clk\_out
  - EthernetReader\_SX\_Cfg, [16](#)
- cnt
  - CommTestSettStruct, [9](#)
- comm\_test.c
  - ctest\_answer\_detected, [40](#)
  - ctest\_answer\_tout, [41](#)
  - ctest\_data\_generate, [41](#)
  - ctest\_get\_result, [41](#)
  - ctest\_handle, [42](#)
  - ctest\_send, [42](#)
  - ctest\_start, [42](#)
  - ctest\_stop, [42](#)
- comm\_test.h
  - ctest\_answer\_detected, [44](#)
  - ctest\_answer\_tout, [44](#)
  - ctest\_get\_result, [44](#)
  - ctest\_handle, [45](#)
  - ctest\_start, [45](#)
  - ctest\_stop, [45](#)
- CommStruct, [6](#)

- addr\_dest, 6
- answer\_id, 6
- answer\_timeout\_flag, 6
- answer\_tmr, 6
- awaiting\_answer, 7
- send\_tries, 7
- sent\_request, 7
- source\_dev, 7
- CommTestResultStruct, 7
  - completed, 8
  - in\_progress, 8
  - nok, 8
  - ok, 8
  - reserved, 8
- CommTestSettStruct, 9
  - addr, 9
  - cnt, 9
  - data\_len, 9
  - timeout, 9
- completed
  - CommTestResultStruct, 8
- config.h
  - USE\_USB\_PLL, 49
- conn
  - Leds\_struct, 20
- conn\_blink
  - Leds\_struct, 20
- ctest\_answer\_detected
  - comm\_test.c, 40
  - comm\_test.h, 44
- ctest\_answer\_tout
  - comm\_test.c, 41
  - comm\_test.h, 44
- ctest\_data\_generate
  - comm\_test.c, 41
- ctest\_get\_result
  - comm\_test.c, 41
  - comm\_test.h, 44
- ctest\_handle
  - comm\_test.c, 42
  - comm\_test.h, 45
- ctest\_send
  - comm\_test.c, 42
- ctest\_start
  - comm\_test.c, 42
  - comm\_test.h, 45
- ctest\_stop
  - comm\_test.c, 42
  - comm\_test.h, 45
- data\_len
  - CommTestSettStruct, 9
- dev\_broadcast\_handle
  - extcomm.c, 52
- dev\_handle
  - extcomm.c, 53
- device\_load\_cfg\_defaults
  - cfg.c, 32
- device\_load\_configuration
  - cfg.c, 32
- dw\_load\_cfg\_defaults
  - cfg.c, 33
- dw\_load\_configuration
  - cfg.c, 33
- dw\_select
  - modules.c, 74
- dw\_unselect
  - modules.c, 74
- e\_cfg\_type
  - ARM\_Ethernet\_Reader.h, 30
- e\_chrg\_mode
  - power\_ctrl.h, 105
- e\_module
  - modules.h, 82
- e\_module\_type
  - modules.h, 83
- e\_st\_type
  - ARM\_Ethernet\_Reader.h, 31
- err
  - Leds\_struct, 20
- ErrorHandler
  - nanotron.c, 87
- EthernetReader\_Cfg, 10
  - header, 10
  - module1\_on, 10
  - module2\_on, 10
  - nanotron\_clk\_en, 10
  - reserved, 11
  - route\_from\_wireless1, 11
  - route\_from\_wireless2, 11
  - usb\_enabled, 11
- EthernetReader\_DW\_Cfg, 11
  - header, 12
- EthernetReader\_NTR\_Cfg, 12
  - header, 12
  - mode, 13
  - nano\_pwr, 13
- EthernetReader\_SX\_Cfg, 16
  - clk\_out, 16
  - header, 16
  - if\_gain, 17
  - listen\_on\_chn2, 17
  - reserved, 17
  - rf\_frequency\_1, 17
  - rf\_frequency\_2, 17
  - rf\_power, 17
  - vco\_trim, 17
- EthernetReader\_State, 13
  - batt\_voltage, 14
  - charging, 14
  - chrg\_current, 14
  - chrg\_mode, 14
  - header, 14
  - module1\_rssi\_back, 14
  - module1\_type, 14
  - module2\_rssi\_back, 15
  - module2\_type, 15

- pwr\_good, 15
- reserved, 15
- server\_connected, 15
- supply\_voltage, 15
- temperature, 15
- uptime, 15
- usb\_voltage, 16
- extcomm.c
  - answer\_handle, 51
  - answer\_ms\_tick, 51
  - answer\_timeout\_handle, 52
  - beacon\_broadcast, 52
  - clear\_answer\_timer, 52
  - dev\_broadcast\_handle, 52
  - dev\_handle, 53
  - is\_dev\_handle, 53
  - module1\_handle, 54
  - module2\_handle, 54
  - packet\_route, 54
  - send\_wakeup\_burst, 55
  - set\_answer\_timer, 55
  - sx1\_pck\_handle, 56
  - sx2\_pck\_handle, 56
  - tcp\_pck\_handle, 56
  - uart\_sys\_pck\_handle, 56
  - udp\_pck\_handle, 57
  - usb\_pck\_handle, 57
- extcomm.h
  - answer\_ms\_tick, 58
  - answer\_timeout\_handle, 59
  - beacon\_broadcast, 59
  - clear\_answer\_timer, 59
  - module1\_handle, 59
  - module2\_handle, 60
  - send\_wakeup\_burst, 60
  - set\_answer\_timer, 61
  - tcp\_pck\_handle, 61
  - uart\_sys\_pck\_handle, 61
  - udp\_pck\_handle, 61
  - usb\_pck\_handle, 62
- fix\_version
  - cfg.c, 33
  - cfg.h, 37
- Flag\_struct, 18
  - ms100, 18
  - power\_off, 18
  - reinit\_modules, 18
  - sec, 19
  - sx1\_irq\_detected, 19
  - sx2\_irq\_detected, 19
  - update\_cfg, 19
  - usb\_forced, 19
  - wakeup\_burst, 19
- GPIO\_IRQHandler
  - main.c, 64
- handle\_led
  - main.h, 68
- handle\_led\_blink
  - main.h, 68
- handle\_timer
  - main.h, 70
- handle\_timer\_flags
  - main.h, 70
- header
  - EthernetReader\_Cfg, 10
  - EthernetReader\_DW\_Cfg, 12
  - EthernetReader\_NTR\_Cfg, 12
  - EthernetReader\_SX\_Cfg, 16
  - EthernetReader\_State, 14
  - Module\_SX\_Cfg, 23
- if\_gain
  - EthernetReader\_SX\_Cfg, 17
- in\_progress
  - CommTestResultStruct, 8
- interface
  - ROUTING\_TABLE\_ITEM, 24
- is\_dev\_handle
  - extcomm.c, 53
- Leds\_struct, 20
  - conn, 20
  - conn\_blink, 20
  - err, 20
  - mod1, 20
  - mod1\_blink, 21
  - mod2, 21
  - mod2\_blink, 21
  - pwr, 21
  - reserved, 21
  - sys, 21
- lib/common/ARM\_Ethernet\_Reader.h, 29
- listen\_on\_chn2
  - EthernetReader\_SX\_Cfg, 17
- load\_configuration
  - cfg.c, 34
  - cfg.h, 37
- MODULE\_STRUCT, 22
  - slot1\_type, 22
  - slot2\_type, 22
- main
  - main.c, 65
  - main\_bootloader.c, 72
- main.c
  - ADC\_init, 64
  - Chip\_USB\_Init, 64
  - GPIO\_IRQHandler, 64
  - main, 65
  - read\_system\_timer, 65
  - reinit\_SX\_modules, 65
  - SysTick\_Handler, 66
  - SystemClkOut\_init, 65
  - SystemCoreClock\_init, 66
- main.h

- Chip\_USB\_Init, 71
  - handle\_led, 68
  - handle\_led\_blink, 68
  - handle\_timer, 70
  - handle\_timer\_flags, 70
  - read\_system\_timer, 71
  - SystemClkOut\_init, 71
- main\_bootloader.c
  - main, 72
- mod1
  - Leds\_struct, 20
- mod1\_blink
  - Leds\_struct, 21
- mod2
  - Leds\_struct, 21
- mod2\_blink
  - Leds\_struct, 21
- mode
  - EthernetReader\_NTR\_Cfg, 13
- module1\_handle
  - extcomm.c, 54
  - extcomm.h, 59
- module1\_on
  - EthernetReader\_Cfg, 10
- module1\_rssi\_back
  - EthernetReader\_State, 14
- module1\_type
  - EthernetReader\_State, 14
- module2\_handle
  - extcomm.c, 54
  - extcomm.h, 60
- module2\_on
  - EthernetReader\_Cfg, 10
- module2\_rssi\_back
  - EthernetReader\_State, 15
- module2\_type
  - EthernetReader\_State, 15
- Module\_SX\_Cfg, 22
  - header, 23
  - vco\_trim, 23
- module\_configure\_pins
  - modules.c, 77
- module\_detect
  - modules.c, 77
- module\_dw\_present
  - modules.c, 77
- module\_get\_rssi
  - modules.c, 78
  - modules.h, 83
- module\_get\_rssi\_background
  - modules.c, 78
  - modules.h, 83
- module\_ntr\_present
  - modules.c, 79
- module\_pck\_send
  - modules.c, 79
  - modules.h, 84
- module\_pck\_send\_receive
  - modules.c, 79
  - modules.h, 84
- module\_reset
  - modules.c, 75
- module\_set\_pwr
  - modules.c, 80
- module\_set\_sleep
  - modules.c, 80
- module\_sx\_present
  - modules.c, 81
- ModuleCfg, 23
- modules.c
  - dw\_select, 74
  - dw\_unselect, 74
  - module\_configure\_pins, 77
  - module\_detect, 77
  - module\_dw\_present, 77
  - module\_get\_rssi, 78
  - module\_get\_rssi\_background, 78
  - module\_ntr\_present, 79
  - module\_pck\_send, 79
  - module\_pck\_send\_receive, 79
  - module\_reset, 75
  - module\_set\_pwr, 80
  - module\_set\_sleep, 80
  - module\_sx\_present, 81
  - modules\_init, 81
  - ntr\_select, 75
  - ntr\_unselect, 76
  - sx\_select, 76
  - sx\_unselect, 76
- modules.h
  - e\_module, 82
  - e\_module\_type, 83
  - module\_get\_rssi, 83
  - module\_get\_rssi\_background, 83
  - module\_pck\_send, 84
  - module\_pck\_send\_receive, 84
  - modules\_init, 85
- modules\_init
  - modules.c, 81
  - modules.h, 85
- ms100
  - Flag\_struct, 18
  - Timer\_struct, 26
- nano\_pwr
  - EthernetReader\_NTR\_Cfg, 13
- nanotron.c
  - ErrorHandler, 87
  - nanotron\_beacon, 87
  - nanotron\_broadcast\_rtc, 87
  - nanotron\_calibrate, 88
  - nanotron\_calibrate\_all, 88
  - nanotron\_enable\_promiscuous, 88
  - nanotron\_get\_ptr, 89
  - nanotron\_get\_toa, 89
  - nanotron\_init, 89
  - nanotron\_init\_all, 90

- nanotron\_pwr\_down, 90
- nanotron\_rx\_enable, 90
- nanotron\_set\_tx\_pwr, 91
- nanotron\_task, 91
- send\_timesync\_packet, 91
- nanotron.h
  - nanotron\_beacon, 93
  - nanotron\_broadcast\_rtc, 94
  - nanotron\_calibrate, 94
  - nanotron\_calibrate\_all, 94
  - nanotron\_enable\_promiscuous, 95
  - nanotron\_get\_ptr, 95
  - nanotron\_get\_toa, 95
  - nanotron\_init, 96
  - nanotron\_init\_all, 96
  - nanotron\_rx\_enable, 96
  - nanotron\_set\_tx\_pwr, 97
  - nanotron\_task, 97
  - SendMsg, 93
- nanotron\_beacon
  - nanotron.c, 87
  - nanotron.h, 93
- nanotron\_broadcast\_rtc
  - nanotron.c, 87
  - nanotron.h, 94
- nanotron\_calibrate
  - nanotron.c, 88
  - nanotron.h, 94
- nanotron\_calibrate\_all
  - nanotron.c, 88
  - nanotron.h, 94
- nanotron\_clk\_en
  - EthernetReader\_Cfg, 10
- nanotron\_enable\_promiscuous
  - nanotron.c, 88
  - nanotron.h, 95
- nanotron\_get\_ptr
  - nanotron.c, 89
  - nanotron.h, 95
- nanotron\_get\_toa
  - nanotron.c, 89
  - nanotron.h, 95
- nanotron\_init
  - nanotron.c, 89
  - nanotron.h, 96
- nanotron\_init\_all
  - nanotron.c, 90
  - nanotron.h, 96
- nanotron\_pwr\_down
  - nanotron.c, 90
- nanotron\_rx\_enable
  - nanotron.c, 90
  - nanotron.h, 96
- nanotron\_set\_tx\_pwr
  - nanotron.c, 91
  - nanotron.h, 97
- nanotron\_task
  - nanotron.c, 91
- nanotron.h, 97
- nok
  - CommTestResultStruct, 8
- ntr\_load\_cfg\_defaults
  - cfg.c, 34
- ntr\_load\_configuration
  - cfg.c, 35
- ntr\_select
  - modules.c, 75
- ntr\_unselect
  - modules.c, 76
- ok
  - CommTestResultStruct, 8
- packet\_route
  - extcomm.c, 54
- pindef.c
  - pindef\_init, 98
- pindef.h
  - pindef\_init, 102
- pindef\_init
  - pindef.c, 98
  - pindef.h, 102
- power\_ctrl.c
  - avg\_handle, 103
  - pwr\_handle, 103
  - pwr\_indication, 103
  - pwr\_meas\_vals, 104
  - pwr\_suicide, 104
  - set\_chrg\_mode, 104
- power\_ctrl.h
  - e\_chrg\_mode, 105
  - pwr\_handle, 106
  - pwr\_meas\_vals, 106
  - pwr\_suicide, 106
  - set\_chrg\_mode, 106
- power\_off
  - Flag\_struct, 18
- press\_time
  - Timer\_struct, 27
- pwr
  - Leds\_struct, 21
- pwr\_good
  - EthernetReader\_State, 15
- pwr\_handle
  - power\_ctrl.c, 103
  - power\_ctrl.h, 106
- pwr\_indication
  - power\_ctrl.c, 103
- pwr\_meas\_vals
  - power\_ctrl.c, 104
  - power\_ctrl.h, 106
- pwr\_suicide
  - power\_ctrl.c, 104
  - power\_ctrl.h, 106
- ROUTING\_TABLE\_ITEM, 24
  - addr, 24

- interface, 24
- timer, 24
- read\_system\_timer
  - main.c, 65
  - main.h, 71
- reinit\_SX\_modules
  - main.c, 65
- reinit\_modules
  - Flag\_struct, 18
- reserved
  - CommTestResultStruct, 8
  - EthernetReader\_Cfg, 11
  - EthernetReader\_SX\_Cfg, 17
  - EthernetReader\_State, 15
  - Leds\_struct, 21
- rf\_frequency\_1
  - EthernetReader\_SX\_Cfg, 17
- rf\_frequency\_2
  - EthernetReader\_SX\_Cfg, 17
- rf\_power
  - EthernetReader\_SX\_Cfg, 17
- route\_from\_wireless1
  - EthernetReader\_Cfg, 11
- route\_from\_wireless2
  - EthernetReader\_Cfg, 11
- routing.c
  - routing\_table\_device\_del, 108
  - routing\_table\_find, 109
  - routing\_table\_timeout, 109
  - routing\_table\_update, 109
- routing.h
  - routing\_table\_device\_del, 111
  - routing\_table\_find, 111
  - routing\_table\_timeout, 112
  - routing\_table\_update, 112
- routing\_table\_device\_del
  - routing.c, 108
  - routing.h, 111
- routing\_table\_find
  - routing.c, 109
  - routing.h, 111
- routing\_table\_timeout
  - routing.c, 109
  - routing.h, 112
- routing\_table\_update
  - routing.c, 109
  - routing.h, 112
- rss
  - rss\_i\_table\_t, 25
- rss.c
  - rss\_i\_table\_add, 113
  - rss\_i\_table\_get, 113
  - rss\_i\_table\_tick, 114
- rss.h
  - rss\_i\_table\_add, 115
  - rss\_i\_table\_get, 115
  - rss\_i\_table\_tick, 116
- rss\_i\_table\_add
  - rss.c, 113
  - rss.h, 115
- rss\_i\_table\_get
  - rss.c, 113
  - rss.h, 115
- rss\_i\_table\_t, 25
  - addr, 25
  - rss, 25
  - timer, 25
- rss\_i\_table\_tick
  - rss.c, 114
  - rss.h, 116
- SPI\_Master\_Init
  - spi\_wrapper.c, 117
  - spi\_wrapper.h, 119
- SPI\_Read\_Write
  - spi\_wrapper.c, 117
  - spi\_wrapper.h, 120
- SPI\_Wait\_Busy
  - spi\_wrapper.c, 118
  - spi\_wrapper.h, 120
- SPI\_Write
  - spi\_wrapper.c, 118
  - spi\_wrapper.h, 120
- samples
  - AvgStruct, 5
- save\_configuration
  - cfg.c, 35
  - cfg.h, 39
- sec
  - Flag\_struct, 19
  - Timer\_struct, 27
- send\_timesync\_packet
  - nanotron.c, 91
- send\_tries
  - CommStruct, 7
- send\_wakeup\_burst
  - extcomm.c, 55
  - extcomm.h, 60
- SendMsg
  - nanotron.h, 93
- sent\_request
  - CommStruct, 7
- server\_connected
  - EthernetReader\_State, 15
- set\_answer\_timer
  - extcomm.c, 55
  - extcomm.h, 61
- set\_chrg\_mode
  - power\_ctrl.c, 104
  - power\_ctrl.h, 106
- slot1\_type
  - MODULE\_STRUCT, 22
- slot2\_type
  - MODULE\_STRUCT, 22
- source\_dev
  - CommStruct, 7
- spi\_wrapper.c



- SPI\_Master\_Init, 117
- SPI\_Read\_Write, 117
- SPI\_Wait\_Busy, 118
- SPI\_Write, 118
- spi\_wrapper.h
  - SPI\_Master\_Init, 119
  - SPI\_Read\_Write, 120
  - SPI\_Wait\_Busy, 120
  - SPI\_Write, 120
- src/cfg.c, 31
- src/cfg.h, 36
- src/comm\_test.c, 39
- src/comm\_test.h, 43
- src/config.h, 46
- src/extcomm.c, 49
- src/extcomm.h, 57
- src/hardfault.c, 62
- src/main.c, 62
- src/main.h, 66
- src/main\_bootloader.c, 72
- src/modules.c, 73
- src/modules.h, 81
- src/nanotron.c, 85
- src/nanotron.h, 92
- src/pindef.c, 97
- src/pindef.h, 98
- src/power\_ctrl.c, 102
- src/power\_ctrl.h, 105
- src/routing.c, 107
- src/routing.h, 110
- src/rssi.c, 112
- src/rssi.h, 114
- src/spi\_wrapper.c, 116
- src/spi\_wrapper.h, 118
- sum
  - AvgStruct, 5
- supply\_voltage
  - EthernetReader\_State, 15
- sx1\_irq\_detected
  - Flag\_struct, 19
- sx1\_pck\_handle
  - extcomm.c, 56
- sx2\_irq\_detected
  - Flag\_struct, 19
- sx2\_pck\_handle
  - extcomm.c, 56
- sx\_load\_cfg\_defaults
  - cfg.c, 35
- sx\_load\_configuration
  - cfg.c, 36
- sx\_select
  - modules.c, 76
- sx\_unselect
  - modules.c, 76
- sys
  - Leds\_struct, 21
- SysTick\_Handler
  - main.c, 66
- SystemClkOut\_init
  - main.c, 65
  - main.h, 71
- SystemCoreClock\_init
  - main.c, 66
- TOA\_DATA, 27
- tcp\_pck\_handle
  - extcomm.c, 56
  - extcomm.h, 61
- temperature
  - EthernetReader\_State, 15
- timeout
  - CommTestSettStruct, 9
- timer
  - ROUTING\_TABLE\_ITEM, 24
  - rssi\_table\_t, 25
- timer32
  - Timer\_struct, 27
- Timer\_struct, 26
  - bootloader, 26
  - button, 26
  - buzz, 26
  - ms100, 26
  - press\_time, 27
  - sec, 27
  - timer32, 27
  - timesync\_broadcast, 27
- timesync\_broadcast
  - Timer\_struct, 27
- USE\_USB\_PLL
  - config.h, 49
- uart\_sys\_pck\_handle
  - extcomm.c, 56
  - extcomm.h, 61
- udp\_pck\_handle
  - extcomm.c, 57
  - extcomm.h, 61
- update\_cfg
  - Flag\_struct, 19
- update\_configuration
  - cfg.c, 36
  - cfg.h, 39
- uptime
  - EthernetReader\_State, 15
- usb\_enabled
  - EthernetReader\_Cfg, 11
- usb\_forced
  - Flag\_struct, 19
- usb\_pck\_handle
  - extcomm.c, 57
  - extcomm.h, 62
- usb\_voltage
  - EthernetReader\_State, 16
- vco\_trim
  - EthernetReader\_SX\_Cfg, 17
  - Module\_SX\_Cfg, 23

wakeup\_burst  
    Flag\_struct, [19](#)