Ethernet Reader

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

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 ℃]

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 requets 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

CFG_DEVICE	Device configuration
CFG_MODULE1_SX	Semtech module 1 configuration
CFG_MODULE1_NTR	Nanotron module 1 configuration
CFG_MODULE1_DW	Decawave module 1 configuration
CFG_MODULE2_SX	Semtech module 2 configuration
CFG_MODULE2_NTR	Nanotron module 2 configuration
CFG_MODULE2_DW	Decawave module 2 configuration
CFG_ALL	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

module	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	*p_cfg*	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	*p_cfg*	Pointer to Decawave configuration
	addr_eeprom	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

header	Pointer to header of configuration
cfg_type	Type of configuration which will be fixed
cfg_ver	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

cfg_type	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	*p_cfg*	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	*p_cfg*	Pointer to Nanotron configuration
	addr_eeprom	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

cfg_type	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	*p_cfg*	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	*p_cfg*	Pointer to Semtech configuration
	addr_eeprom	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

header	Pointer to header of configuration
cfg_type	Type of configuration which will be fixed
cfg_ver	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

cfg_type	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

cfg_type	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	*pck*	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

buffer	Pointer to buffer for saving data
len	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

out	*buff*	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	*pck*	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

	source_device	Device peripheral which received packet (USB, UART, etc.)
	receive_status	Status of received packet
in	*pck*	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

	source_device	Device peripheral which received packet (USB, UART, etc.)
	receive_status	Status of received packet
	pck_in	Incoming packet
out	*pck_out*	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

	source_device	Device peripheral which received packet (USB, UART, etc.)
	receive_status	Status of received packet
in,out	*pck*	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

	source_device	Device peripheral which received packet (USB, UART, etc.)
	receive_status	Status of received packet
in,out	*pck*	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

	source_device	Device peripheral which received packet (USB, UART, etc.)
	pck_in	Incoming packet
out	*pck_out*	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

	src_addr	Address of source device
	dest_addr	Address of destination device
	count	Number of packet in burst
	rq	Request send in packets
	data_len	Length of data send in packet
in	*data*	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

timeout	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

	src_addr	Address of source device
	dest_addr	Address of destination device
	count	Number of packet in burst
	rq	Request send in packets
	data_len	Length of data send in packet
in	*data*	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

timeout	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

enabled	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

var	LED lighting timer
led	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

tmr	LED lighting timer
blink	Perform LED blink
invert	Inverts LED blinking
led	LED pin definition

4.11.2.3 handle_timer

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

Value:

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

Handling timer decrement.

Parameters

timer	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

tmr	Handled timer
val	Value loaded to timer when reaches zero
flag	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

enabled	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

module	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

module	Module index
state	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

module	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

module	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

module	Module index
type	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

module	Module index

Returns

Nothing

4.13.3.3 module_dw_present()

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

Decawave module detection.

Parameters

module	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

module	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

module	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

module	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

	module	Module index
in	*pck*	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

	module	Module index
in	*pckin*	Pointer to packet which will be send through selected module
out	*pckout*	Pointer to packet buffer where will be put received packet
	timeout	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

module	Module index
pwr_state	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

module	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

module	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

module	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

module	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

	module	Module index
in	*pck*	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

	module	Module index
in	*pckin*	Pointer to packet which will be send through selected module
out	*pckout*	Pointer to packet buffer where will be put received packet
	timeout	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 <nnspi.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_##i##_CS(0)
- #define **NTRX_UNSELECT**(i) NTRX_##i##_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

err	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

ngX	Pointer to NanoLOC work & configuration structure

Returns

Nothing

4.15.2.4 nanotron_calibrate()

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

Calibrate one NanoLOC module.

Parameters

ngX	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

index	NanoLOC chip index
enabled	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

index	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

ngX	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

index	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

ngX	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

ngX	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

ngX	Pointer to NanoLOC work & configuration structure
pwr	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

ngX	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

ngX	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

ngX	Pointer to NanoLOC work & configuration structure

Returns

Nothing

4.16.3.3 nanotron_calibrate()

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

Calibrate one NanoLOC module.

Parameters

ngX	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

index	NanoLOC chip index
enabled	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

index	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

ngX	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

index	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

ngX	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

ngX	Pointer to NanoLOC work & configuration structure
pwr	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

ngX	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_PLLLOCK1_DIR**(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 3, 25, val)
- #define **SX_PLLLOCK1** 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_PLLLOCK2_DIR**(val) Chip_GPIO_WriteDirBit(LPC_GPIO, 3, 26, val)
- #define **SX_PLLLOCK2** 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

avg_t	Pointer to averaging structure
in_val	Input value for averaging
result	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

mode	Current mode. Valid values avaible in enum e_chrg_mode

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

mode	Current mode. Valid values avaible 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

addr	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

addr	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

addr	Sorce device address
interface	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

addr	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

addr	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

addr	Sorce device address
interface	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

addr	Device address
module	Module which received packet
rssi	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

buffer	Pointer to buffer where RSSI table will be copied
index	Index in RSSI table where the copying starts
max_rec_cnt	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

addr	Device address
module	Module which received packet
rssi	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

buffer	Pointer to buffer where RSSI table will be copied
index	Index in RSSI table where the copying starts
max_rec_cnt	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

index	Index of SPI peripheral
freq	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

index	Index of SPI peripheral
data	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

index	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

index	Index of SPI peripheral
data	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

index	Index of SPI peripheral
freq	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

index	Index of SPI peripheral
data	Data to send

Returns

Read data

Parameters

index	Index of SPI peripheral
data	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

index	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

index	Index of SPI peripheral
data	Data to send

Returns

Nothing

Index