Gateway

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 BorderRouter Class Reference

Public Member Functions

- **BorderRouter** (const BorderRouter &)
- void **operator=** (const BorderRouter &)
- eBorderRouterStatus init (wpanManager ∗_wpan)

  init function
- void deInit ()

  deInit
- eBorderRouterStatus reInit ()

  deInit then init @reutnr own status
- eBorderRouterStatus getStatus ()

  status getter
- void tick (uint32_t delayMs)

  master ticker with delay

Static Public Member Functions

- static BorderRouter & getInstance ()

  singleton get instance

4.1.1 Member Function Documentation

4.1.1.1 getStatus()

```
eBorderRouterStatus BorderRouter::getStatus ( )
```

status getter

Returns

border router status

4.1.1.2 init()

```
eBorderRouterStatus BorderRouter::init (
            wpanManager * _wpan )
```

init function

Returns

own status WPAN manager must be initialized elsewhere

4.1.1.3 tick()

```
void BorderRouter::tick (
            uint32_t delayMs )
```

master ticker with delay

Parameters

delay	in ms Ticks WPAN manager and all Connections. Updates node lists and Connections if a new DataNode was connected

The documentation for this class was generated from the following files:

- Router/BorderRouter.h
- Router/BorderRouter.cpp

4.2 Connection Class Reference

Public Member Functions

- Connection (DataNode ∗_pNode)

  Constructor.
- ∼Connection ()

  Destructor De-init used sockets.
- void tick ()

  check Connection IO check for new messages from socket and radio module Call communication handlers if new data must be transferred
- int getNodeAddr ()

  getter function
- eConnectionState getStatus ()

  getter function

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Connection()

```
Connection::Connection (
            DataNode * _pNode )
```

Constructor.

Parameters

pointer	to target dataNode

4.2.2 Member Function Documentation

4.2.2.1 getNodeAddr()

```
int Connection::getNodeAddr ( )
```

getter function

Returns

node address

4.2.2.2 getStatus()

```
eConnectionState Connection::getStatus ( )
```

getter function

Returns

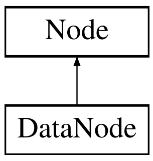
status enum value

The documentation for this class was generated from the following files:

- Router/Connection.h
- Router/Connection.cpp

4.3 DataNode Class Reference

Inheritance diagram for DataNode:



Public Member Functions

- DataNode ()

  Default constructor Init with address, group and lease duration = 0.
- DataNode (uint8_t _address, uint32_t _group)

  Static constructor.
- DataNode (uint8_t _address, uint32_t _group, uint8_t _leaseDuration)

  Dynamic node constructor.
- DataNode (Node &base)

  Constructor from existing Node.
- ∼DataNode ()

  Destructor Clears RX datagrams and frees their allocated memory.
- void show ()

  printer function
- uint16_t datagram_tx (uint8_t ∗data, uint16_t dataSize)

  TX datagram to node.
- int readyRxDatagrams ()
- uint16_t readDatagram (uint8_t ∗buffer, uint16_t maxSize)

  read first ready frame
- void clearDatagrams ()

  Clear datagrams and free their allocated memory.

Additional Inherited Members

4.3.1 Constructor & Destructor Documentation

4.3.1.1 DataNode() `[1/3]`

```
DataNode::DataNode (
            uint8_t _address,
            uint32_t _group )
```

Static constructor.

Parameters

address	and group Init with arguments and lease duration = 0

4.3.1.2 DataNode() [2/3]

```
DataNode::DataNode (
            uint8_t _address,
            uint32_t _group,
            uint8_t _leaseDuration )
```

Dynamic node constructor.

Parameters

address	and group
lease	duration (multiple of NETWORK_LEASE_UNIT_MINUTES)

4.3.1.3 DataNode() [3/3]

```
DataNode::DataNode (
            Node & base )
```

Constructor from existing Node.

Parameters

Node	to transtype Copy argument node address, group and lease duration. Do not copy current callback flags

4.3.2 Member Function Documentation

4.3.2.1 datagram_tx()

```
uint16_t DataNode::datagram_tx (
            uint8_t * data,
            uint16_t dataSize )
```

TX datagram to node.

Parameters

data	and dataSize Will not transmit if dataSize>DATA_MAX_DATAGRAM_LENGTH. Will segment into multiple packets if data is too big for one ISM frame

4.3.2.2 readDatagram()

```
uint16_t DataNode::readDatagram (
            uint8_t * buffer,
            uint16_t maxSize )
```

read first ready frame

Parameters

buffer	to fill with data
buffer	max size

Returns

size of filled data

4.3.2.3 readyRxDatagrams()

```
int DataNode::readyRxDatagrams ( )
```

Returns

number of received complete datagrams

The documentation for this class was generated from the following files:

- Gateway/datanode.h
- Gateway/datanode.cpp

4.4 ism_stat_t Struct Reference

Public Attributes

- uint32_t **rxOk**
- uint32_t **rxCrcError**
- uint32_t **tx**
- uint32_t **txLbtFail**

- uint32_t **txAck**
- uint32_t **txNack**
- uint32_t **syncRxOk**
- uint32_t **syncRxCrcError**
- uint32_t **syncRxBadFrame**
- uint32_t **syncRxTimeout**
- uint32_t **syncRxLost**
- int16_t **syncMinDelta**
- int16_t **syncMaxDelta**
- int32_t **syncSumDelta**
- uint32_t **lpsyncRxOk**
- uint32_t **lpsyncRxCrcError**
- uint32_t **lpsyncRxBadFrame**
- uint32_t **lpsyncRxTimeout**
- uint32_t **lpsyncRxLost**
- int16_t **lpsyncMinDelta**
- int16_t **lpsyncMaxDelta**
- int32_t **lpsyncSumDelta**
- uint32_t **syncTxOk**
- uint32_t **syncTxLbtFail**
- uint32_t **rxScanTime**
- uint32_t **rxTime**
- uint32_t **txTime**
- uint32_t **txUnicast**
- uint32_t **txUnicastAck**
- uint32_t **txUnicastNack**
- uint32_t **txMulticast**
- uint32_t **txMulticastAttempt**
- uint32_t **rxUnicastOk**
- uint32_t **rxMulticastOk**
- uint32_t **rxBadFrame**
- uint32_t **rxWrongBeaconId**
- uint32_t **rxUnicastDuplicate**
- uint32_t **rxUnicastWrongAddress**
- uint32_t **rxMulticastWrongGroup**
- uint32_t **scanOnPhase**
- uint32_t **scanLock**
- uint32_t **scanLockTimeout**
- uint32_t **scanLockRssiTooLow**
- uint32_t **scanRxCrcError**
- uint32_t **scanRxBadFrame**
- uint32_t **scanRefuseUnassociated**
- uint32_t **scanSuccess**

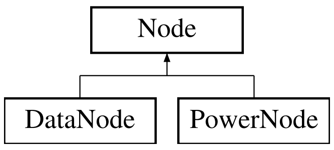
The documentation for this struct was generated from the following file:

- ISM3_Linux/ism3.h

4.5 Node Class Reference

```
#include <node.h>
```

Inheritance diagram for Node:



Public Member Functions

- Node ()

  Default constructor.
- Node (uint8_t _address, uint32_t _group)

  Static definition constructor.
- Node (uint8_t _address, uint32_t _group, uint8_t _leaseDuration)

  Dynamic definition constructor.
- virtual ∼Node ()

  Destructor.
- virtual void show ()

  printer function
- uint8_t getUid (uint8_t ∗buffer, uint8_t size)

  get unique device id
- uint8_t **getAddr** ()
- uint8_t getOldAddr ()

  get address before address was changed
- uint32_t getGroup ()

  get node group
- uint8_t getLeaseDuration ()

  get lease duration
- uint32_t getLeaseStartTime ()

  get lease start time
- uint8_t getProtocols ()

  get announced supported network protocols
- virtual uint8_t getNodeTypeProtocols ()

  get class-supported network protocols
- bool wakeup ()

  Wake low power group the node is part of.
- bool sleep ()

  Unwake low power group the node is part of.
- bool net_ping (uint32_t timeoutMs)
- bool net_getUid (uint32_t timeoutMs)
- bool net_setAddr (uint8_t newAddr)
- bool net_setAddrAgain (uint8_t maxTries, uint32_t timeoutMs)
- bool net_setGroup (uint32_t newGroup)
- bool net_disconnect (uint32_t timeoutMs)

  disconnects Node

- bool net_getProtocols (uint32_t timeoutMs)

  Get node supported protocols.
- bool pingStatus ()
- bool uidStatus ()
- bool protocolsStatus ()
- bool isConnected ()
- bool isLeaseValid ()
- uint8_t tx (uint8_t ∗buffer, uint8_t length)

  Send buffer to Node using unicast frame.
- void rxCallback (const uint8_t ∗data, uint8_t size)

  master RX handler, calls corresponding protocol handler

Protected Member Functions

- bool txTimeout (uint8_t ∗frame, uint8_t length, uint32_t timeoutMs, bool ∗callbackFlag)

  TX frame expecting arg flag to be raised, with timeout.

Protected Attributes

- uint8_t **protocols** =0
- uint32_t **leaseStartTime** =0

Friends

- ostream & **operator**<< (ostream &out, const Node &node)

4.5.1 Detailed Description

Node class as seen by the network controller. DO NOT CONFUSE WITH ACTUAL NODE, this class is only a soft representation

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Node() [1/2]

```
Node::Node (
            uint8_t _address,
            uint32_t _group )
```

Static definition constructor.

Parameters

node	address
node	group

4.5.2.2 Node() [2/2]

```
Node::Node (
            uint8_t _address,
            uint32_t _group,
            uint8_t _leaseDuration )
```

Dynamic definition constructor.

Parameters

node	address
node	group
lease	duration

4.5.3 Member Function Documentation

4.5.3.1 getGroup()

```
uint32_t Node::getGroup ( )
```

get node group

Returns

group

4.5.3.2 getLeaseDuration()

```
uint8_t Node::getLeaseDuration ( )
```

get lease duration

Returns

lease duration

4.5.3.3 getLeaseStartTime()

```
uint32_t Node::getLeaseStartTime ( )
```

get lease start time

Returns

lease start time in UNIX epoch seconds

4.5.3.4 getNodeTypeProtocols()

```
uint8_t Node::getNodeTypeProtocols ( ) [virtual]
```

get class-supported network protocols

Returns

protocol bitfield

Reimplemented in PowerNode.

4.5.3.5 getOldAddr()

```
uint8_t Node::getOldAddr ( )
```

get address before address was changed

Returns

previous address Useful if address was not changed as expected

4.5.3.6 getProtocols()

```
uint8_t Node::getProtocols ( )
```

get announced supported network protocols

Returns

protocol bitfield

4.5.3.7 getUid()

```
uint8_t Node::getUid (
             uint8_t * buffer,
             uint8_t size )
```

get unique device id

Parameters

buffer	to store the uid in
size	of buffer, must be > NODE_UID8_WIDTH

Returns

number of bytes written to buffer

4.5.3.8 isConnected()

```
bool Node::isConnected ( )
```

Returns

connection status

4.5.3.9 isLeaseValid()

```
bool Node::isLeaseValid ( )
```

Returns

true if lease is not expired or static

4.5.3.10 net_disconnect()

```
bool Node::net_disconnect (
             uint32_t timeoutMs )
```

disconnects Node

Parameters

timeout	in ms

Returns

true if disconnect was confirmed within timeout

4.5.3.11 net_getProtocols()

```
bool Node::net_getProtocols (
            uint32_t timeoutMs )
```

Get node supported protocols.

Parameters

timeout	in ms

Returns

true if protocols were fetched within timeout

4.5.3.12 net_getUid()

```
bool Node::net_getUid (
            uint32_t timeoutMs )
```

Parameters

timeout	in ms

Returns

boolean uid get result If timeout is 0, function is non-blocking and returns false. In this case, use uidStatus to check command result

4.5.3.13 net_ping()

```
bool Node::net_ping (
            uint32_t timeoutMs )
```

Parameters

timeout	in ms

Returns

ping result If timeout is 0, function is non-blocking and returns false. In this case, use pingStatus to check ping operation

4.5.3.14 net_setAddr()

```
bool Node::net_setAddr (
            uint8_t newAddr )
```

Parameters

new	address

Returns

always true This command is not confirmed, manual confirmation through ping is needed

4.5.3.15 net_setAddrAgain()

```
bool Node::net_setAddrAgain (
            uint8_t maxTries,
            uint32_t timeoutMs )
```

Returns

always true

Parameters

maxTries	max number of new net_setAddr commands
timeoutMs	timeout for EACH try in

Returns

true if address was successfully changed Set node address again using net_setAddr to previous node address Ping to confirm new address

4.5.3.16 net_setGroup()

```
bool Node::net_setGroup (
            uint32_t newGroup )
```

Parameters

new	group

Returns

always true This command is not confirmed, manual confirmation through ping is needed

4.5.3.17 pingStatus()

```
bool Node::pingStatus ( )
```

Returns

ping result

4.5.3.18 protocolsStatus()

```
bool Node::protocolsStatus ( )
```

Returns

get protocol cmd status

4.5.3.19 rxCallback()

```
void Node::rxCallback (
             const uint8_t * data,
             uint8_t size )
```

master RX handler, calls corresponding protocol handler

Parameters

frame	data and size from lower layer callback

Returns

none

4.5.3.20 sleep()

```
bool Node::sleep ( )
```

Unwake low power group the node is part of.

Returns

true if unwake command was sent

4.5.3.21 tx()

```
uint8_t Node::tx (
            uint8_t * buffer,
            uint8_t length )
```

Send buffer to Node using unicast frame.

Parameters

unsigned	char buffer
buffer	length

4.5.3.22 txTimeout()

```
bool Node::txTimeout (
            uint8_t * frame,
            uint8_t length,
            uint32_t timeoutMs,
            bool * callbackFlag )  [protected]
```

TX frame expecting arg flag to be raised, with timeout.

Parameters

frame	and frame length
timeout	in ms
flag	to be set by RX callback function

4.5.3.23 uidStatus()

```
bool Node::uidStatus ( )
```

Returns

uid get cmd status

4.5.3.24 wakeup()

```
bool Node::wakeup ( )
```

Wake low power group the node is part of.

Returns

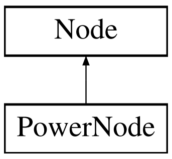
true if wakeup command was sent

The documentation for this class was generated from the following files:

- Gateway/node.h
- Gateway/node.cpp

4.6 PowerNode Class Reference

Inheritance diagram for PowerNode:



Public Member Functions

- PowerNode ()

  Default constructor Init with address, group and lease duration = 0.
- PowerNode (uint8_t _address, uint32_t _group)

  Static definition constructor.
- PowerNode (uint8_t _address, uint32_t _group, uint8_t _leaseDuration)

  Dynamic node constructor.
- PowerNode (Node &base)

  Constructor from existing Node.
- ∼PowerNode ()

  Destructor.
- void show ()

  printer function
- vector< powerTarget_t > **getPowerSettingsList** ()
- powerkW_t **getPower** ()
- powerSetting_t **getPowerSetting** ()
- sManifest **getManifest** ()
- string **getDescription** ()
- uint8_t getNodeTypeProtocols ()

  get announced supported network protocols
- bool app_getPowerSettings (uint32_t timeoutMs)

  get power setting list from node
- bool app_getPower (uint32_t timeoutMs)

get current node power as float
- bool app_setPower (powerkW_t powerkW_t, uint32_t timeoutMs)

  set node power as float
- bool app_getPowerSetting (uint32_t timeoutMs)

  get current node power setting
- bool app_setPowerSetting (powerSetting_t powerSetting_t, uint32_t timeoutMs)

  set current node power setting
- bool app_getManifest (uint32_t timeoutMs)

  get node manifest
- bool **getManifestStatus** ()
- bool **setPowerStatus** ()
- bool **getPowerStatus** ()
- bool **setPowerSettingStatus** ()
- bool **getPowerSettingStatus** ()
- bool **getPowerSettingsStatus** ()

Friends

- ostream & **operator**<< (ostream &out, const PowerNode &powerNode)

Additional Inherited Members

4.6.1 Constructor & Destructor Documentation

4.6.1.1 PowerNode() [1/3]

```
PowerNode::PowerNode (
             uint8_t _address,
             uint32_t _group )
```

Static definition constructor.

Parameters

address	and group Init with arguments and lease duration = 0

4.6.1.2 PowerNode() [2/3]

```
PowerNode::PowerNode (
             uint8_t _address,
             uint32_t _group,
             uint8_t _leaseDuration )
```

Dynamic node constructor.

Parameters

address	and group
lease	duration (multiple of NETWORK_LEASE_UNIT_MINUTES)

4.6.1.3 PowerNode() `[3/3]`

```
PowerNode::PowerNode (
             Node & base )
```

Constructor from existing Node.

Parameters

Node	to transtype Copy argument node address, group and lease duration. Do not copy current callback flags

4.6.2 Member Function Documentation

4.6.2.1 app_getManifest()

```
bool PowerNode::app_getManifest (
             uint32_t timeoutMs )
```

get node manifest

Parameters

timeout	in ms, no timeout if 0

Returns

true if power was received within timeout, 0 otherwise

4.6.2.2 app_getPower()

```
bool PowerNode::app_getPower (
             uint32_t timeoutMs )
```

get current node power as float

Parameters

timeout	in ms, no timeout if 0

Returns

true if power was received within timeout, 0 otherwise

4.6.2.3 app_getPowerSetting()

```
bool PowerNode::app_getPowerSetting (
             uint32_t timeoutMs )
```

get current node power setting

Parameters

timeout	in ms, no timeout if 0

Returns

true if power setting was received within timeout, 0 otherwise

4.6.2.4 app_getPowerSettings()

```
bool PowerNode::app_getPowerSettings (
             uint32_t timeoutMs )
```

get power setting list from node

Parameters

timeout	in ms, no timeout if 0

Returns

true if power settings got within timeout, 0 otherwise

4.6.2.5 app_setPower()

```
bool PowerNode::app_setPower (
             powerkW_t powerkW_t,
             uint32_t timeoutMs )
```

set node power as float

Parameters

timeout	in ms, no timeout if 0
target	power

Returns

true if power set ack got within timeout, 0 otherwise Applicability depends on distant node implementation (will return error APP_ERR_UNDEFINEDCMD)

4.6.2.6 app_setPowerSetting()

```
bool PowerNode::app_setPowerSetting (
            powerSetting_t powerSetting_t,
            uint32_t timeoutMs )
```

set current node power setting

Parameters

timeout	in ms, no timeout if 0
new	power setting index

Returns

true if power setting ack was received within timeout, 0 otherwise

4.6.2.7 getNodeTypeProtocols()

```
uint8_t PowerNode::getNodeTypeProtocols ( )  [virtual]
```

get announced supported network protocols

Returns

protocol bitfield

Reimplemented from Node.

The documentation for this class was generated from the following files:

- Gateway/powernode.h
- Gateway/powernode.cpp

4.7 sDatagram Struct Reference

Public Attributes

- bool **ready**
- datagram_id_t **id**
- uint16_t **size**
- uint8_t * **data**

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

- Protocol/wpan.h

4.8 sManifest Struct Reference

Public Attributes

- uint8_t **priority**
- uint8_t **descriptionLength**
- char * **description**

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

- Protocol/wpan.h

4.9 sOffer Struct Reference

Offer storage data coupling.

```
#include <wpanManager.h>
```

Public Attributes

- uint8_t **address**
- uint8_t **uidHash**

4.9.1 Detailed Description

Offer storage data coupling.

sOffer allows storage of a pair of address and UID hash. It is used to store DORA offers

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

- Gateway/wpanManager.h

4.10 sPowerSettings Struct Reference

Public Attributes

- uint8_t **nPowerSettings**
- uint8_t **powerSettingWidth**
- powerTarget_t ∗ **powerSettingskW**

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

- Protocol/wpan.h

4.11 uartParam_s Struct Reference

Public Attributes

- parity_t **parity**
- uint32_t **word_length**
- flowControl_t **flow_control**
- stopBits_t **stopBits**
- uint32_t **baudrate**

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

- ISM3_Linux/buffered_uart.c

4.12 wpanManager Class Reference

Master handler of radio module and connected nodes.

```
#include <wpanManager.h>
```

Public Member Functions

- wpanManager ()

  Default constructor Init with default server power and no static nodes.
- wpanManager (vector< Node > nodeList_)

  Static address list constructor.
- wpanManager (uint8_t power_, uint8_t power_dbm_)

  Power setting constructor.
- wpanManager (vector< Node > nodeList_, uint8_t power_, uint8_t power_dbm_)

  Complete constructor.
- vector< Node ∗ > getNodeList ()
- vector< Node ∗ > getStaticNodeList ()
- vector< PowerNode ∗ > getPowerNodeList ()
- vector< DataNode ∗ > getDataNodeList ()

- bool nodeListUpdated ()
- void clearNodeLists ()

  Clear all node lists Send disconnect command to all nodes to notify them.
- void clearDynamicNodeList ()

  Clear dynamically-addressed nodes Send disconnect command to deleted nodes to notify them.
- void clearStaticNodeList ()

  Clear statically-addressed nodes Send disconnect command to deleted nodes to notify them.
- void tick (uint32_t delayMs_)

  ticker for WPAN manager
- void startDynamicDiscovery ()

  Start dynamic discovery Wake up dynamic discovery group.
- void stopDynamicDiscovery ()

  Stop dynamic discovery Put dynamic discovery group to sleep.
- void updateNodeTypes ()

  update node types from nodes list Ask non-identified base nodes for their supported protocols.
- void printNodes ()

  print node list Prints statically and dynamically-addressed nodes
- void printStaticNodes ()

  print static node list Prints statically-addressed nodes
- void rxHandler (const uint8_t ∗data, uint8_t size, uint8_t source)

  handle RX data

4.12.1 Detailed Description

Master handler of radio module and connected nodes.

This class is a multi-part master handler for a WPAN.

It handles:

- radio module initialization in gateway mode
- radio module ticks
- various node lists
- dynamic address definition
- packet reception and subsequent dispatching to node class instances for processing

4.12.2 module initialization: WPAN manager uses ism3_server.c/h to configure

gateway radio at desired or default power.

Radio module ticks: WPAN manager tick implements a sleep-delayed tick function that calls the ISM3 driver tick function.

Node lists: There are several overlapping node lists. As of now, WPAN manager maintains 2∗n separate lists, n being the number of different node types (Base, Data, Power). Statically and dynamically-addressed nodes are kept in different node lists. This implementation should be changed to a node type -agnostic implementation to simplify code and execution. Handles radio module init, ticks, address attribution and packet redirection to client node handlers (Node class and sub-classes). Exports a list of connected nodes to be used elsewhere

4.12.3 Constructor & Destructor Documentation

4.12.3.1 wpanManager() [1/3]

```
wpanManager::wpanManager (
            vector< Node > nodeList_ )
```

Static address list constructor.

Parameters

Static	node vector Init with default server power

4.12.3.2 wpanManager() [2/3]

```
wpanManager::wpanManager (
            uint8_t power_,
            uint8_t power_dbm_ )
```

Power setting constructor.

Parameters

Power	setting and matching dBm Init with custom server power and no static nodes

4.12.3.3 wpanManager() [3/3]

```
wpanManager::wpanManager (
            vector< Node > nodeList_,
            uint8_t power_,
            uint8_t power_dbm_ )
```

Complete constructor.

Parameters

Static	node vector
Power	setting and matching dBm

4.12.4 Member Function Documentation

4.12.4.1 getDataNodeList()

```
vector< DataNode * > wpanManager::getDataNodeList ( )
```

Returns

Vector of pointers to all data nodes

4.12.4.2 getNodeList()

```
vector< Node * > wpanManager::getNodeList ( )
```

Returns

Vector of pointers to all nodes

4.12.4.3 getPowerNodeList()

```
vector< PowerNode * > wpanManager::getPowerNodeList ( )
```

Returns

Vector of pointers to all power nodes

4.12.4.4 getStaticNodeList()

```
vector< Node * > wpanManager::getStaticNodeList ( )
```

Returns

Vector of pointers to all static nodes

4.12.4.5 nodeListUpdated()

```
bool wpanManager::nodeListUpdated ( )
```

Returns

True if node list has changed since last poll

4.12.4.6 rxHandler()

```
void wpanManager::rxHandler (
             const uint8_t * data,
             uint8_t size,
             uint8_t source )
```

handle RX data

Parameters

RX	data and size
data	source Call relevant callback if source is a registered node. Call DORA handler on DORA command reception

4.12.4.7 tick()

```
void wpanManager::tick (
            uint32_t delayMs_ )
```

ticker for WPAN manager

Parameters

tick	delay in ms Handles ISM tick, periodic node list update and lease expiry checks Tick delay is implemented here as sleep period

The documentation for this class was generated from the following files:

- Gateway/wpanManager.h
- Gateway/wpanManager.cpp

Chapter 5

File Documentation

5.1 ISM3_Linux/buffered_uart.c File Reference

Driver for UART using circular buffer.

```
#include <assert.h>
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <sys/signal.h>
#include <sys/types.h>
#include "buffered_uart.h"
#include "hardware.h"
```

Classes

- struct uartParam_s

Macros

- #define **RX_BUFFER_SIZE** 512
- #define **TX_BUFFER_SIZE** 1024
- #define **N_BAUD_SETTINGS** 22

Functions

- void **buffered_uart_init** (uint32_t baudrate, uint32_t word_length, parity_t parity, flowControl_t flow_control)
- void **buffered_uart_deinit** (void)
- void **buffered_uart_start** (void)
- void **buffered_uart_set_baudrate** (uint32_t baudrate)
- uint16_t **buffered_uart_read** (uint8_t ∗data, uint16_t size)
- uint16_t **buffered_uart_write** (const uint8_t ∗data, uint16_t size)
- uint16_t **buffered_uart_get_read_available** (void)
- uint16_t **buffered_uart_get_write_available** (void)
- bool **buffered_uart_is_busy** (void)

Variables

- uartParam_s **uConfig**

5.1.1 Detailed Description

Driver for UART using circular buffer.

Author

nicolas.brunner@heig-vd.ch

Date

07-August-2018

Copyright

HEIG-VD

License information

5.2 ISM3_Linux/buffered_uart.h File Reference

Driver for UART using circular buffer.

```
#include <stdbool.h>
#include <stdint.h>
#include "hardware.h"
```

Enumerations

- enum **parity_t** { **PARITY_NONE** , **PARITY_EVEN** , **PARITY_ODD** }
- enum **stopBits_t** { **STOP_BITS_1** , **STOP_BITS_2** }
- enum **flowControl_t** { **FLOW_CONTROL_NONE** , **FLOW_CONTROL_RTS_CTS** }

Functions

- void **buffered_uart_init** (uint32_t baudrate, uint32_t word_length, parity_t parity, flowControl_t flow_control)
- void **buffered_uart_deinit** (void)
- void **buffered_uart_start** (void)
- void **buffered_uart_set_baudrate** (uint32_t baudrate)
- uint16_t **buffered_uart_read** (uint8_t ∗data, uint16_t size)
- uint16_t **buffered_uart_write** (const uint8_t ∗data, uint16_t size)
- uint16_t **buffered_uart_get_read_available** (void)
- uint16_t **buffered_uart_get_write_available** (void)
- bool **buffered_uart_is_busy** (void)

5.2.1 Detailed Description

Driver for UART using circular buffer.

Author

nicolas.brunner@heig-vd.ch

Date

07-August-2018

Copyright

HEIG-VD

License information

5.3 ISM3_Linux/framed_uart.c File Reference

Driver for UART using frame.

```
#include <assert.h>
#include <time.h>
#include <stdio.h>
#include "buffered_uart.h"
#include "framed_uart.h"
```

Macros

- #define **RX_MAX_FRAME_SIZE** 256

Functions

- void **framed_uart_init** (framed_uart_function_t rx_function_, uint32_t frame_timeout_)
- void **framed_uart_deinit** (void)
- void **framed_uart_start** (void)
- bool **framed_uart_tx** (const uint8_t *data, uint16_t size)
- void **framed_uart_set_baudrate** (uint32_t baudrate)
- void **framed_uart_flush** (void)
- void **framed_uart_tick** (void)
- bool **framed_uart_is_busy** (void)

5.3.1 Detailed Description

Driver for UART using frame.

Author

nicolas.brunner@heig-vd.ch

Date

06-August-2018

Copyright

HEIG-VD

License information

5.4 ISM3_Linux/framed_uart.h File Reference

Driver for UART using frame.

```
#include <stdbool.h>
#include <stdint.h>
```

Typedefs

- typedef void(∗ **framed_uart_function_t**) (const uint8_t ∗data, uint16_t size)

Functions

- void **framed_uart_init** (framed_uart_function_t rx_function, uint32_t frame_timeout)
- void **framed_uart_deinit** (void)
- void **framed_uart_start** (void)
- void **framed_uart_set_baudrate** (uint32_t baudrate)
- void **framed_uart_flush** (void)
- void **framed_uart_tick** (void)
- bool **framed_uart_tx** (const uint8_t ∗data, uint16_t size)
- bool **framed_uart_is_busy** (void)

5.4.1 Detailed Description

Driver for UART using frame.

Author

nicolas.brunner@heig-vd.ch

Date

06-August-2018

Copyright

HEIG-VD

License information

5.5 ISM3_Linux/ism3.c File Reference

Driver for the RM1S3.

```
#include <assert.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "commands_RM1S3.h"
#include "framed_uart.h"
#include "ism3.h"
#include "hardware.h"
#include "util.h"
```

Macros

- #define **RESET_DURATION** 10
- #define **START_DURATION** 500
- #define **FRAME_TIMEOUT** 150
- #define **BAUDRATE_CHANGE_DURATION** 5
- #define **BAUDRATE** 19200
- #define **TX_HEADER_SIZE** 4
- #define **BROADCAST_HEADER_SIZE** TX_HEADER_SIZE+4
- #define **RX_HEADER_SIZE** 6
- #define **SOURCE_INDEX** 2
- #define **DATA_SLOT_INDEX** 3
- #define **RSSI_INDEX** 4
- #define **LQI_INDEX** 5
- #define **RX_MULTICAST_HEADER_SIZE** 11
- #define **RX_MULTICAST_SOURCE_INDEX** 2
- #define **RX_MULTICAST_GROUP_INDEX** 3

- #define **RX_MULTICAST_COUNTDOWN_INDEX** 7
- #define **RX_MULTICAST_RSSI_INDEX** 9
- #define **RX_MULTICAST_LQI_INDEX** 10
- #define **GROUP_SIZE** 4
- #define **MAX_COMMAND_SIZE** 18
- #define **NUMBER_OF_RECONFIGURATION_CMD** 9
- #define **STATE_UNINITIALIZED** 0xFF
- #define **TX_STATUS_NONE** 0
- #define **TX_STATUS_WAIT_ACK** 1
- #define **TX_STATUS_ACK** 2
- #define **TX_STATUS_TIMEOUT** 3
- #define **DEFAULT_PHY1** 0
- #define **DEFAULT_PHY2** 3
- #define **UNALLOWED_PHY1** 1
- #define **UNALLOWED_PHY2** 2
- #define **DEFAULT_POWER** 0x06
- #define **DEFAULT_POWER_DBM** 14
- #define **NUMBER_OF_GPIO** 9

Typedefs

- typedef const uint8_t **commands_t**[ ][MAX_COMMAND_SIZE]

Functions

- void ism_init (ism_unicast_function_t rx_unicast_function_, ism_multicast_function_t rx_multicast_function←_, ism_beacon_data_function_t beacon_data_function_, ism_state_function_t state_function_, ism_stat_←function_t stat_function_)
- void **ism_config** (uint8_t address_, uint32_t group_, uint8_t power_, uint8_t power_dbm_, uint64_←t associated_beacon_id_)
- void **ism_get_config** (uint8_t ∗address_, uint32_t ∗group_, uint8_t ∗power_, uint8_t ∗power_dbm_, uint64←_t ∗associated_beacon_id_)
- void **ism_get_uid** (uint8_t ∗uid_, uint8_t uid_size_)
- bool ism_set_phy (uint8_t phy_, const uint8_t ∗channels_)
- void **ism_disconnect** (void)
- void **ism_set_sync_mode** (ism_sync_mode_t mode)
- void **ism_power_down** (void)
- void **ism_tick** (void)
- bool **ism_is_tx_pending** (void)
- bool **ism_is_ready** (void)
- bool **ism_tx** (uint8_t destination, const uint8_t ∗data, uint8_t size)
- bool **ism_broadcast** (uint32_t group, uint8_t number, const uint8_t ∗data, uint8_t size)
- uint8_t **ism_get_max_data_size** (void)
- char ∗ **ism_get_firmware_version** (void)
- uint32_t ism_get_firmware_version_value (void)
- char ∗ **ism_get_hardware_version** (void)
- bool **ism_request_stat** (void)
- bool **ism_request_state** (void)
- bool **ism_update_firmware** (const uint8_t ∗firmware, uint32_t size)
- uint32_t **ism_get_uart_rx_counter** (void)
- uint8_t **ism_get_channels_size** (uint8_t phy)
- bool **send_command** (const uint8_t ∗data)
- void **EXTI15_10_IRQHandler** (void)

5.5.1 Detailed Description

Driver for the RM1S3.

Author

nicolas.brunner@heig-vd.ch

Date

06-August-2018

Copyright

HEIG-VD

License information

5.5.2 Function Documentation

5.5.2.1 ism_get_firmware_version_value()

```
uint32_t ism_get_firmware_version_value (
             void   )
```

Get the firmware version coded into an integer version a.b.c => a << 16 + b << 8 + c

Returns

the firmware version coded into an integer

5.5.2.2 ism_init()

```
void ism_init (
             ism_unicast_function_t rx_unicast_function,
             ism_multicast_function_t rx_multicast_function,
             ism_beacon_data_function_t beacon_data_function,
             ism_state_function_t state_function,
             ism_stat_function_t stat_function )
```

Initialize the ISM

Parameters

rx_unicast_function	the function called when unicast are received
rx_multicast_function	the function called when multicast data are received
beacon_data_function	the function called when beacon data are received
state_function	the function called when the state change
stat_function	the function called when stat are read

5.5.2.3 ism_set_phy()

```
bool ism_set_phy (
            uint8_t phy,
            const uint8_t * channels )
```

Set the physical layer parameters, they will be use only after a ism_disconnect()

5.6 ISM3_Linux/ism3.h File Reference

Driver for the RM1S3.

```
#include <stdbool.h>
#include <stdint.h>
```

Classes

- struct ism_stat_t

Macros

- #define **ISM_TIMESLOT_DURATION** 20
- #define **ISM_MAX_DATA_SIZE** 239
- #define **ISM_INVALID_POWER** 0xFF
- #define **ISM_MAX_POWER** 52
- #define **ISM_MAX_POWER_DBM** 30
- #define **UID_SIZE** 12

Typedefs

- typedef void(∗ **ism_unicast_function_t**) (const uint8_t ∗data, uint8_t size, uint8_t source, int8_t rssi, uint8↵_t lqi)
- typedef void(∗ **ism_multicast_function_t**) (const uint8_t ∗data, uint8_t size, uint8_t source, uint8_t countdown, int8_t rssi, uint8_t lqi)
- typedef void(∗ **ism_beacon_data_function_t**) (const uint8_t ∗data, uint8_t size)
- typedef void(∗ **ism_state_function_t**) (ism_state_t state, const uint8_t ∗gateway_id)
- typedef void(∗ **ism_stat_function_t**) (ism_stat_t stat)

Enumerations

- enum **ism_state_t** {
  ISM_OFF , **ISM_NOT_SYNCHRONIZED** , **ISM_SYNCHRONIZED** , **ISM_LOW_POWER_SYNC** ,
  ISM_TX_SYNC , **ISM_VERSION_READY** }
- enum **ism_sync_mode_t** { **SM_TX** , **SM_RX_ACTIVE** , **SM_RX_LOW_POWER** , **SM_RX_LOW_POWER↩ _GROUP** }

Functions

- void ism_init (ism_unicast_function_t rx_unicast_function, ism_multicast_function_t rx_multicast_function, ism_beacon_data_function_t beacon_data_function, ism_state_function_t state_function, ism_stat_↩ function_t stat_function)
- void **ism_config** (uint8_t address, uint32_t group, uint8_t power, uint8_t power_dbm, uint64_t associated↩ _beacon_id)
- void **ism_get_config** (uint8_t ∗address, uint32_t ∗group, uint8_t ∗power, uint8_t ∗power_dbm, uint64_↩ t ∗associated_beacon_id)
- void **ism_get_uid** (uint8_t ∗uid_, uint8_t uid_size_)
- bool ism_set_phy (uint8_t phy, const uint8_t ∗channels)
- void **ism_disconnect** (void)
- void **ism_set_sync_mode** (ism_sync_mode_t mode)
- void **ism_tick** (void)
- bool **ism_tx** (uint8_t destination, const uint8_t ∗data, uint8_t size)
- bool **ism_broadcast** (uint32_t group, uint8_t number, const uint8_t ∗data, uint8_t size)
- bool **ism_is_tx_pending** (void)
- bool **ism_is_ready** (void)
- uint8_t **ism_get_max_data_size** (void)
- char ∗ **ism_get_firmware_version** (void)
- uint32_t ism_get_firmware_version_value (void)
- char ∗ **ism_get_hardware_version** (void)
- bool **ism_update_firmware** (const uint8_t ∗firmware, uint32_t size)
- bool **ism_request_stat** (void)
- bool **ism_request_state** (void)
- uint32_t **ism_get_uart_rx_counter** (void)
- uint8_t **ism_get_channels_size** (uint8_t phy)
- bool **send_command** (const uint8_t ∗data)

5.6.1 Detailed Description

Driver for the RM1S3.

Author

nicolas.brunner@heig-vd.ch

Date

06-August-2018

Copyright

HEIG-VD

License information

5.6.2 Function Documentation

5.6.2.1 ism_get_firmware_version_value()

```
uint32_t ism_get_firmware_version_value (
            void )
```

Get the firmware version coded into an integer version a.b.c => a << 16 + b << 8 + c

Returns

the firmware version coded into an integer

5.6.2.2 ism_init()

```
void ism_init (
            ism_unicast_function_t rx_unicast_function,
            ism_multicast_function_t rx_multicast_function,
            ism_beacon_data_function_t beacon_data_function,
            ism_state_function_t state_function,
            ism_stat_function_t stat_function )
```

Initialize the ISM

Parameters

rx_unicast_function	the function called when unicast are received
rx_multicast_function	the function called when multicast data are received
beacon_data_function	the function called when beacon data are received
state_function	the function called when the state change
stat_function	the function called when stat are read

5.6.2.3 ism_set_phy()

```
bool ism_set_phy (
            uint8_t phy,
            const uint8_t * channels )
```

Set the physical layer parameters, they will be use only after a ism_disconnect()

5.7 ISM3_Linux/util.c File Reference

Utility library, use big endian.

```
#include <stdbool.h>
#include "util.h"
```

Functions

- uint8_t **util_get_number_of_bit_set** (uint8_t value)

5.7.1 Detailed Description

Utility library, use big endian.

Author

nicolas.brunner@heig-vd.ch & laurent.folladore@heig-vd.ch

Date

06-September-2012

Copyright

HEIG-VD

License information

5.8 ISM3_Linux/util.h File Reference

Utility library, use big endian.

```
#include <stdint.h>
```

Macros

- #define **UTIL_MIN**(x, y) ((x) < (y) ? (x) : (y))
- #define **UTIL_MAX**(x, y) ((x) > (y) ? (x) : (y))
- #define **UTIL_CEILING**(x, y) (((x) + (y) - 1) / (y))

Functions

- uint8_t **util_get_number_of_bit_set** (uint8_t value)

5.8.1 Detailed Description

Utility library, use big endian.

Author

nicolas.brunner@heig-vd.ch & laurent.folladore@heig-vd.ch

Date

06-September-2012

Copyright

HEIG-VD

License information

Index