WSNet : Wireless Network Simulator

Generated by Doxygen 1.7.6.1

Thu Feb 23 2012 11:25:34

Contents

Chapter 1

Deprecated List

Global get_entity_links_down (call_t *c)
Should use get_entity_bindings_down() instead.

Global get_entity_links_down_nbr (call_t *c)
Should use get_entity_bindings_down() instead.

Global get_entity_links_up (call_t *c)
Should use get_entity_bindings_up() instead.

Global get_entity_links_up_nbr (call_t *c)
Should use get_entity_bindings_up() instead.

Chapter 2

Directory Hierarchy

2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Directory Documentation

5.1 include/ Directory Reference

Files

- file antenna.h

  Antenna declarations.
- file battery.h

  Battery declarations.
- file das.h

  DAta Structure module declarations.
- file dbg.h

  dbg declarations
- file entity.h

  Entity declarations.
- file hadas.h

  Hashed DAta Structure module declarations.
- file ioctl_message.h
- file log.h

  log declarations
- file measure.h

  Measure declarations.
- file medium.h

  Medium declarations.
- file mem_fs.h

  Fixed size memory management module declarations.
- file models.h

  Models declarations.
- file modelutils.h

  Utility function declarations.

- file modulation.h

  Modulation declarations.
- file monitor.h

  Monitor declarations.
- file node.h

  Node declarations.
- file options.h

  User options declarations.
- file packet.h

  Packet declarations.
- file param.h
- file probabilistic_distribution.h

  Probabilistic distributions declarations.
- file radio.h

  Radio declarations.
- file rng.h

  Random number generator declarations.
- file scheduler.h

  Scheduler declarations.
- file sodas.h

  SOrted DAta Structure module declarations.
- file spadas.h

  Space PArtitioning DAta Structure module declarations.
- file timer.h

  Generic timer.
- file types.h

  Type declarations.
- file worldsens_debug.h
- file worldsens_pkt.h

  Worldsens packet format.

Chapter 6

Data Structure Documentation

6.1 _angle Struct Reference

An angle in the 3D space.

`#include <types.h>`

Data Fields

- double xy

  angle on the xy plane
- double z

  angle between the xy plane and the z axis

6.1.1 Detailed Description

An angle in the 3D space.

Should use type angle_t.

Definition at line 82 of file types.h.

6.1.2 Field Documentation

6.1.2.1 double _angle::xy

angle on the xy plane

Definition at line 83 of file types.h.

6.1.2.2 double _angle::z

angle between the xy plane and the z axis

Definition at line 84 of file types.h.

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

- types.h

6.2 _antenna_methods Struct Reference

Methods that should be implemented by an antenna model.

`#include <models.h>`

Data Fields

- void(∗ rx)(call_t ∗c, packet_t ∗packet)
- void(∗ cs)(call_t ∗c, packet_t ∗packet)
- double(∗ get_loss)(call_t ∗c)
- angle_t ∗(∗ get_angle)(call_t ∗c)
- void(∗ set_angle)(call_t ∗c, angle_t ∗angle)
- double(∗ gain_tx)(call_t ∗c, position_t ∗pos)
- double(∗ gain_rx)(call_t ∗c, position_t ∗pos)

6.2.1 Detailed Description

Methods that should be implemented by an antenna model.

Should use type antenna_methods_t.

Definition at line 195 of file models.h.

6.2.2 Field Documentation

6.2.2.1 void(∗ _antenna_methods::cs)(call_t ∗c, packet_t ∗packet)

Definition at line 197 of file models.h.

6.2.2.2 double(∗ _antenna_methods::gain_rx)(call_t ∗c, position_t ∗pos)

Definition at line 202 of file models.h.

6.2.2.3 double(∗ _antenna_methods::gain_tx)(call_t ∗c, position_t ∗pos)

Definition at line 201 of file models.h.

6.2.2.4 angle_t*(* _antenna_methods::get_angle)(call_t *c)

Definition at line 199 of file models.h.

6.2.2.5 double(* _antenna_methods::get_loss)(call_t *c)

Definition at line 198 of file models.h.

6.2.2.6 void(* _antenna_methods::rx)(call_t *c, packet_t *packet)

Definition at line 196 of file models.h.

6.2.2.7 void(* _antenna_methods::set_angle)(call_t *c, angle_t *angle)

Definition at line 200 of file models.h.

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

- models.h

6.3 _application_methods Struct Reference

Methods that should be implemented by an application model.

`#include <models.h>`

Data Fields

- void(* rx)(call_t *c, packet_t *packet)

6.3.1 Detailed Description

Methods that should be implemented by an application model.

Should use type application_methods_t.

Definition at line 289 of file models.h.

6.3.2 Field Documentation

6.3.2.1 void(* _application_methods::rx)(call_t *c, packet_t *packet)

Definition at line 290 of file models.h.

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

- models.h

6.4 _array Struct Reference

An array of integers containing its size.

`#include <types.h>`

Data Fields

- int size

  array size
- int * elts

  array elements

6.4.1 Detailed Description

An array of integers containing its size.

Should use type array_t.

Definition at line 21 of file types.h.

6.4.2 Field Documentation

6.4.2.1 int* _array::elts

array elements

Definition at line 23 of file types.h.

6.4.2.2 int _array::size

array size

Definition at line 22 of file types.h.

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

- types.h

6.5 _call Struct Reference

A parameter that identifies who we are calling and who has called us.

`#include <types.h>`

Data Fields

- entityid_t entity

  the called entity id
- nodeid_t node

  the called node id
- entityid_t from

  the entity that made the call

6.5.1 Detailed Description

A parameter that identifies who we are calling and who has called us.

Kind of a self pointer. Should use type call_t.

Definition at line 106 of file types.h.

6.5.2 Field Documentation

6.5.2.1 entityid_t _call::entity

the called entity id

Definition at line 107 of file types.h.

6.5.2.2 entityid_t _call::from

the entity that made the call

Definition at line 109 of file types.h.

6.5.2.3 nodeid_t _call::node

the called node id

Definition at line 108 of file types.h.

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

- types.h

6.6 _destination Struct Reference

A packet destination.

`#include <types.h>`

Data Fields

- nodeid_t id
  the destination node id
- position_t position
  the destination position

6.6.1 Detailed Description

A packet destination.

May be a node address or a geographical position. Should use type destination_t.

Definition at line 94 of file types.h.

6.6.2 Field Documentation

6.6.2.1 nodeid_t _destination::id

the destination node id

Definition at line 95 of file types.h.

6.6.2.2 position_t _destination::position

the destination position

Definition at line 96 of file types.h.

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

- types.h

6.7 _energy_methods Struct Reference

Methods that should be implemented by an energy model.

`#include <models.h>`

Data Fields

- void(* consume_tx)(call_t *c, uint64_t duration, double txdBm)
- void(* consume_rx)(call_t *c, uint64_t duration)
- void(* consume_idle)(call_t *c, uint64_t duration)
- void(* consume)(call_t *c, double energy)
- double(* energy_consumed)(call_t *c)
- double(* energy_remaining)(call_t *c)
- double(* energy_status)(call_t *c)

6.7.1 Detailed Description

Methods that should be implemented by an energy model.

Should use type energy_methods_t.

Definition at line 176 of file models.h.

6.7.2 Field Documentation

6.7.2.1 void(* _energy_methods::consume)(call_t *c, double energy)

Definition at line 180 of file models.h.

6.7.2.2 void(* _energy_methods::consume_idle)(call_t *c, uint64_t duration)

Definition at line 179 of file models.h.

6.7.2.3 void(* _energy_methods::consume_rx)(call_t *c, uint64_t duration)

Definition at line 178 of file models.h.

6.7.2.4 void(* _energy_methods::consume_tx)(call_t *c, uint64_t duration, double txdBm)

Definition at line 177 of file models.h.

6.7.2.5 double(* _energy_methods::energy_consumed)(call_t *c)

Definition at line 181 of file models.h.

6.7.2.6 double(* _energy_methods::energy_remaining)(call_t *c)

Definition at line 182 of file models.h.

6.7.2.7 double(* _energy_methods::energy_status)(call_t *c)

Definition at line 183 of file models.h.

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

- models.h

6.8 _entityid Struct Reference

An entity identifier.

`#include <types.h>`

6.8.1 Detailed Description

An entity identifier.

Should use type entityid_t.

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

- types.h

6.9 _environment_methods Struct Reference

Methods that should be implemented by an environment model.

`#include <models.h>`

Data Fields

- void(∗ read_measure)(call_t ∗c, measureid_t measure, double ∗value)

6.9.1 Detailed Description

Methods that should be implemented by an environment model.

Should use type environment_methods_t.

Definition at line 135 of file models.h.

6.9.2 Field Documentation

6.9.2.1 void(∗ _environment_methods::read_measure)(call_t ∗c, measureid_t measure, double ∗value)

Definition at line 136 of file models.h.

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

- models.h

6.10 _event Struct Reference

A scheduler event.

`#include <scheduler.h>`

Data Fields

- uint64_t clock
- int priority
- int id
- union {
  struct {
  call_t call
  callback_t callback
  void * arg
  } cb
  struct {
  packet_t * packet
  call_t call
  } rx
  nodeid_t nodeid
  } u

6.10.1 Detailed Description

A scheduler event.

Should use type event_t.

Definition at line 19 of file scheduler.h.

6.10.2 Field Documentation

6.10.2.1 void* _event::arg

Definition at line 28 of file scheduler.h.

6.10.2.2 call_t _event::call

Definition at line 26 of file scheduler.h.

6.10.2.3 callback_t _event::callback

Definition at line 27 of file scheduler.h.

6.10.2.4 struct { ... } _event::cb

6.10.2.5 uint64_t _event::clock

Definition at line 20 of file scheduler.h.

6.10.2.6 int _event::id

Definition at line 22 of file scheduler.h.

6.10.2.7 nodeid_t _event::nodeid

Definition at line 34 of file scheduler.h.

6.10.2.8 packet_t* _event::packet

Definition at line 31 of file scheduler.h.

6.10.2.9 int _event::priority

Definition at line 21 of file scheduler.h.

6.10.2.10 struct { ... } _event::rx

6.10.2.11 union { ... } _event::u

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

- scheduler.h

6.11 _fading_methods Struct Reference

Methods that should be implemented by a fading model.

`#include <models.h>`

Data Fields

- double(* fading)(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst, double rxdBm)

6.11.1 Detailed Description

Methods that should be implemented by a fading model.

Should use type fading_methods_t.

Definition at line 69 of file models.h.

6.11.2 Field Documentation

6.11.2.1 double(∗ _fading_methods::fading)(call_t ∗c, packet_t ∗packet, nodeid_t src, nodeid_t dst, double rxdBm)

Definition at line 70 of file models.h.

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

- models.h

6.12 _interferences_methods Struct Reference

Methods that should be implemented by a interferences model.

`#include <models.h>`

Data Fields

- double(∗ interfere)(call_t ∗c, int channel0, int channel1)

6.12.1 Detailed Description

Methods that should be implemented by a interferences model.

Should use type interferences_methods_t.

Definition at line 95 of file models.h.

6.12.2 Field Documentation

6.12.2.1 double(∗ _interferences_methods::interfere)(call_t ∗c, int channel0, int channel1)

Definition at line 96 of file models.h.

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

- models.h

6.13 _io_ctl_message Struct Reference

An ioctl message.

`#include <types.h>`

6.13.1 Detailed Description

An ioctl message.

Should use type ioctl_message_t.

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

- types.h

6.14 _ioctl_message Struct Reference

`#include <types.h>`

Data Fields

- int type
- int size
- int real_size
- void * body

6.14.1 Detailed Description

Definition at line 159 of file types.h.

6.14.2 Field Documentation

6.14.2.1 void* _ioctl_message::body

Definition at line 167 of file types.h.

6.14.2.2 int _ioctl_message::real_size

Definition at line 165 of file types.h.

6.14.2.3 int _ioctl_message::size

Definition at line 163 of file types.h.

6.14.2.4 int _ioctl_message::type

Definition at line 161 of file types.h.

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

- types.h

6.15 _mac_methods Struct Reference

Methods that should be implemented by a mac model.

`#include <models.h>`

Data Fields

- void(* rx)(call_t *c, packet_t *packet)
- void(* tx)(call_t *c, packet_t *packet)
- int(* set_header)(call_t *c, packet_t *packet, destination_t *dst)
- int(* get_header_size)(call_t *c)
- int(* get_header_real_size)(call_t *c)

6.15.1 Detailed Description

Methods that should be implemented by a mac model.

Should use type mac_methods_t.

Definition at line 251 of file models.h.

6.15.2 Field Documentation

6.15.2.1 int(* _mac_methods::get_header_real_size)(call_t *c)

Definition at line 257 of file models.h.

6.15.2.2 int(* _mac_methods::get_header_size)(call_t *c)

Definition at line 255 of file models.h.

6.15.2.3 void(* _mac_methods::rx)(call_t *c, packet_t *packet)

Definition at line 252 of file models.h.

6.15.2.4 int(* _mac_methods::set_header)(call_t *c, packet_t *packet, destination_t *dst)

Definition at line 254 of file models.h.

6.15.2.5 void(* _mac_methods::tx)(call_t *c, packet_t *packet)

Definition at line 253 of file models.h.

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

- models.h

6.16 _measureid Struct Reference

A measure identifier.

`#include <types.h>`

6.16.1 Detailed Description

A measure identifier.

Should use type measureid_t.

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

- types.h

6.17 _mobility_methods Struct Reference

Methods that should be implemented by a mobility model.

`#include <models.h>`

Data Fields

- void(* update_position)(call_t *c)

6.17.1 Detailed Description

Methods that should be implemented by a mobility model.

Should use type mobility_methods_t.

Definition at line 163 of file models.h.

6.17.2 Field Documentation

6.17.2.1 void(* _mobility_methods::update_position)(call_t *c)

Definition at line 164 of file models.h.

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

- models.h

6.18 _model Struct Reference

Information about a model.

`#include <models.h>`

Data Fields

- char * oneline
- char * author
- char * version
- int type
- struct {
  char ** exported
  int count
  } measure

6.18.1 Detailed Description

Information about a model.

Should use type model_t.

Definition at line 36 of file models.h.

6.18.2 Field Documentation

6.18.2.1 char* _model::author

Definition at line 38 of file models.h.

6.18.2.2 int _model::count

Definition at line 43 of file models.h.

6.18.2.3 char** _model::exported

Definition at line 42 of file models.h.

6.18.2.4 struct { ... } _model::measure

6.18.2.5 char* _model::oneline

Definition at line 37 of file models.h.

6.18.2.6 int _model::type

Definition at line 40 of file models.h.

6.18.2.7 char* _model::version

Definition at line 39 of file models.h.

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

- models.h

6.19 _modulation_methods Struct Reference

Methods that should be implemented by a modulation model.

`#include <models.h>`

Data Fields

- double(* modulate)(call_t *c, double snr)
- int(* bit_per_symbol)(call_t *c)

6.19.1 Detailed Description

Methods that should be implemented by a modulation model.

Should use type modulation_methods_t.

Definition at line 121 of file models.h.

6.19.2 Field Documentation

6.19.2.1 int(∗ _modulation_methods::bit_per_symbol)(call_t ∗c)

Definition at line 123 of file models.h.

6.19.2.2 double(∗ _modulation_methods::modulate)(call_t ∗c, double snr)

Definition at line 122 of file models.h.

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

- models.h

6.20 _monitor_methods Struct Reference

Methods that should be implemented by a monitor model.

`#include <models.h>`

Data Fields

- void(∗ monitor_death)(call_t ∗c)
- void(∗ monitor_event)(call_t ∗c)
- void(∗ monitor_register_callback)(call_t ∗c, callback_t callback, void ∗arg)

6.20.1 Detailed Description

Methods that should be implemented by a monitor model.

Should use type monitor_methods_t.

Definition at line 148 of file models.h.

6.20.2 Field Documentation

6.20.2.1 void(∗ _monitor_methods::monitor_death)(call_t ∗c)

Definition at line 149 of file models.h.

6.20.2.2 void(∗ _monitor_methods::monitor_event)(call_t ∗c)

Definition at line 150 of file models.h.

6.20.2.3 void(∗ _monitor_methods::monitor_register_callback)(call_t ∗c, callback_t callback, void ∗arg)

Definition at line 151 of file models.h.

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

- models.h

6.21 _nodeid Struct Reference

A node identifier.

`#include <types.h>`

6.21.1 Detailed Description

A node identifier.

Should use type nodeid_t.

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

- types.h

6.22 _noise_methods Struct Reference

Methods that should be implemented by a noise model.

`#include <models.h>`

Data Fields

- double(∗ noise)(call_t ∗c, nodeid_t node, int channel)

6.22.1 Detailed Description

Methods that should be implemented by a noise model.

Should use type noise_methods_t.

Definition at line 108 of file models.h.

6.22.2 Field Documentation

6.22.2.1 double(* _noise_methods::noise)(call_t *c, nodeid_t node, int channel)

Definition at line 109 of file models.h.

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

- models.h

6.23 _packet Struct Reference

A radio packet.

`#include <types.h>`

Data Fields

- packetid_t id
  the packet id
- int size
  size of the packet data
- int real_size
  real size of the packet data
- int type
  type of the packet (for multistandard nodes)
- uint64_t clock0
  packet rx start
- uint64_t clock1
  packet rx end
- uint64_t duration
  packet tx/rx duration
- nodeid_t node
  node that has created the packet
- entityid_t antenna
  antenna that has emitted the packet
- int worldsens_mod
  worldsens modulation id (wsim radio modulation id)
- double worldsens_freq
  worldsens radio frequency
- double txdBm
  tx power in dBm
- int channel
- entityid_t modulation
  modulation entity
- uint64_t Tb

radio bandwidth: time to send a bit

- double PER

  packet error rate
- double rxdBm

  rx power in dBm
- double rxmW

  rx power in mW
- double * noise_mW

  packet noise in mW
- double * ber

  packet ber
- char * data

  packet data

6.23.1 Detailed Description

A radio packet.

Should use type packet_t.

Definition at line 119 of file types.h.

6.23.2 Field Documentation

6.23.2.1 entityid_t _packet::antenna

antenna that has emitted the packet

Definition at line 130 of file types.h.

6.23.2.2 double* _packet::ber

packet ber

Definition at line 146 of file types.h.

6.23.2.3 int _packet::channel

Definition at line 138 of file types.h.

6.23.2.4 uint64_t _packet::clock0

packet rx start

Definition at line 125 of file types.h.

6.23.2.5 uint64_t _packet::clock1

packet rx end

Definition at line 126 of file types.h.

6.23.2.6 char* _packet::data

packet data

Definition at line 148 of file types.h.

6.23.2.7 uint64_t _packet::duration

packet tx/rx duration

Definition at line 127 of file types.h.

6.23.2.8 packetid_t _packet::id

the packet id

Definition at line 120 of file types.h.

6.23.2.9 entityid_t _packet::modulation

modulation entity

Definition at line 139 of file types.h.

6.23.2.10 nodeid_t _packet::node

node that has created the packet

Definition at line 129 of file types.h.

6.23.2.11 double* _packet::noise_mW

packet noise in mW

Definition at line 145 of file types.h.

6.23.2.12 double _packet::PER

packet error rate

Definition at line 142 of file types.h.

6.23.2.13 int _packet::real_size

real size of the packet data

Definition at line 122 of file types.h.

6.23.2.14 double _packet::rxdBm

rx power in dBm

Definition at line 143 of file types.h.

6.23.2.15 double _packet::rxmW

rx power in mW

Definition at line 144 of file types.h.

6.23.2.16 int _packet::size

size of the packet data

Definition at line 121 of file types.h.

6.23.2.17 uint64_t _packet::Tb

radio bandwidth: time to send a bit

Definition at line 140 of file types.h.

6.23.2.18 double _packet::txdBm

tx power in dBm

Definition at line 137 of file types.h.

6.23.2.19 int _packet::type

type of the packet (for multistandard nodes)

Definition at line 123 of file types.h.

6.23.2.20 double _packet::worldsens_freq

worldsens radio frequency

Definition at line 134 of file types.h.

6.23.2.21 int _packet::worldsens_mod

worldsens modulation id (wsim radio modulation id)

Definition at line 133 of file types.h.

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

- types.h

6.24 _packetid Struct Reference

A packet identifier.

`#include <types.h>`

6.24.1 Detailed Description

A packet identifier.

Should use type packetid_t.

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

- types.h

6.25 _param Struct Reference

A parameter for the "init" and "setnode" entity functions.

`#include <types.h>`

Data Fields

- char * key

  the parameter key
- char * value

  the parameter value

6.25.1 Detailed Description

A parameter for the "init" and "setnode" entity functions.

Should use type param_t.

Definition at line 178 of file types.h.

6.25.2 Field Documentation

6.25.2.1 char* _param::key

the parameter key

Definition at line 179 of file types.h.

6.25.2.2 char* _param::value

the parameter value

Definition at line 180 of file types.h.

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

- types.h

6.26 _position Struct Reference

A position in the 3D space.

`#include <types.h>`

Data Fields

- double x
  x position
- double y
  y position
- double z
  z position

6.26.1 Detailed Description

A position in the 3D space.

Should use type position_t.

Definition at line 69 of file types.h.

6.26.2 Field Documentation

6.26.2.1 double _position::x

x position

Definition at line 70 of file types.h.

6.26.2.2 double _position::y

y position

Definition at line 71 of file types.h.

6.26.2.3 double _position::z

z position

Definition at line 72 of file types.h.

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

- types.h

6.27 _propagation_methods Struct Reference

Methods that should be implemented by a propagation model.

`#include <models.h>`

Data Fields

- double(∗ propagation)(call_t ∗c, packet_t ∗packet, nodeid_t src, nodeid_t dst, double rxdBm)

6.27.1 Detailed Description

Methods that should be implemented by a propagation model.

Should use type propagation_methods_t.

Definition at line 56 of file models.h.

6.27.2 Field Documentation

6.27.2.1 double(∗ _propagation_methods::propagation)(call_t ∗c, packet_t ∗packet, nodeid_t src, nodeid_t dst, double rxdBm)

Definition at line 57 of file models.h.

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

- models.h

6.28 _radio_methods Struct Reference

Methods that should be implemented by a radio model.

`#include <models.h>`

Data Fields

- void(* rx)(call_t *c, packet_t *packet)
- void(* tx)(call_t *c, packet_t *packet)
- int(* set_header)(call_t *c, packet_t *packet, destination_t *dst)
- int(* get_header_size)(call_t *c)
- int(* get_header_real_size)(call_t *c)
- void(* tx_end)(call_t *c, packet_t *packet)
- void(* cs)(call_t *c, packet_t *packet)
- double(* get_noise)(call_t *c)
- double(* get_cs)(call_t *c)
- double(* get_power)(call_t *c)
- void(* set_power)(call_t *c, double power)
- int(* get_channel)(call_t *c)
- void(* set_channel)(call_t *c, int channel)
- entityid_t(* get_modulation)(call_t *c)
- void(* set_modulation)(call_t *c, entityid_t modulation)
- uint64_t(* get_Tb)(call_t *c)
- uint64_t(* get_Ts)(call_t *c)
- void(* set_Ts)(call_t *c, uint64_t Ts)
- double(* get_sensibility)(call_t *c)
- void(* set_sensibility)(call_t *c, double sensibility)
- void(* sleep)(call_t *c)
- void(* wakeup)(call_t *c)
- int(* get_modulation_bit_per_symbol)(call_t *c)

6.28.1 Detailed Description

Methods that should be implemented by a radio model.

Should use type radio_methods_t.

Definition at line 214 of file models.h.

6.28.2 Field Documentation

6.28.2.1 void(* _radio_methods::cs)(call_t *c, packet_t *packet)

Definition at line 223 of file models.h.

6.28.2.2 int(∗ _radio_methods::get_channel)(call_t ∗c)

Definition at line 228 of file models.h.

6.28.2.3 double(∗ _radio_methods::get_cs)(call_t ∗c)

Definition at line 225 of file models.h.

6.28.2.4 int(∗ _radio_methods::get_header_real_size)(call_t ∗c)

Definition at line 220 of file models.h.

6.28.2.5 int(∗ _radio_methods::get_header_size)(call_t ∗c)

Definition at line 219 of file models.h.

6.28.2.6 entityid_t(∗ _radio_methods::get_modulation)(call_t ∗c)

Definition at line 230 of file models.h.

6.28.2.7 int(∗ _radio_methods::get_modulation_bit_per_symbol)(call_t ∗c)

Definition at line 239 of file models.h.

6.28.2.8 double(∗ _radio_methods::get_noise)(call_t ∗c)

Definition at line 224 of file models.h.

6.28.2.9 double(∗ _radio_methods::get_power)(call_t ∗c)

Definition at line 226 of file models.h.

6.28.2.10 double(∗ _radio_methods::get_sensibility)(call_t ∗c)

Definition at line 235 of file models.h.

6.28.2.11 uint64_t(∗ _radio_methods::get_Tb)(call_t ∗c)

Definition at line 232 of file models.h.

6.28.2.12 uint64_t(* _radio_methods::get_Ts)(call_t *c)

Definition at line 233 of file models.h.

6.28.2.13 void(* _radio_methods::rx)(call_t *c, packet_t *packet)

Definition at line 215 of file models.h.

6.28.2.14 void(* _radio_methods::set_channel)(call_t *c, int channel)

Definition at line 229 of file models.h.

6.28.2.15 int(* _radio_methods::set_header)(call_t *c, packet_t *packet, destination_t *dst)

Definition at line 218 of file models.h.

6.28.2.16 void(* _radio_methods::set_modulation)(call_t *c, entityid_t modulation)

Definition at line 231 of file models.h.

6.28.2.17 void(* _radio_methods::set_power)(call_t *c, double power)

Definition at line 227 of file models.h.

6.28.2.18 void(* _radio_methods::set_sensibility)(call_t *c, double sensibility)

Definition at line 236 of file models.h.

6.28.2.19 void(* _radio_methods::set_Ts)(call_t *c, uint64_t Ts)

Definition at line 234 of file models.h.

6.28.2.20 void(* _radio_methods::sleep)(call_t *c)

Definition at line 237 of file models.h.

6.28.2.21 void(* _radio_methods::tx)(call_t *c, packet_t *packet)

Definition at line 216 of file models.h.

6.28.2.22 void(* _radio_methods::tx_end)(call_t *c, packet_t *packet)

Definition at line 222 of file models.h.

6.28.2.23 void(* _radio_methods::wakeup)(call_t *c)

Definition at line 238 of file models.h.

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

- models.h

6.29 _routing_methods Struct Reference

Methods that should be implemented by a routing model.

`#include <models.h>`

Data Fields

- void(* rx)(call_t *c, packet_t *packet)
- void(* tx)(call_t *c, packet_t *packet)
- int(* set_header)(call_t *c, packet_t *packet, destination_t *dst)
- int(* get_header_size)(call_t *c)
- int(* get_header_real_size)(call_t *c)

6.29.1 Detailed Description

Methods that should be implemented by a routing model.

Should use type routing_methods_t.

Definition at line 270 of file models.h.

6.29.2 Field Documentation

6.29.2.1 int(* _routing_methods::get_header_real_size)(call_t *c)

Definition at line 276 of file models.h.

6.29.2.2 int(* _routing_methods::get_header_size)(call_t *c)

Definition at line 274 of file models.h.

6.29.2.3 void(* _routing_methods::rx)(call_t *c, packet_t *packet)

Definition at line 271 of file models.h.

6.29.2.4 int(* _routing_methods::set_header)(call_t *c, packet_t *packet, destination_t *dst)

Definition at line 273 of file models.h.

6.29.2.5 void(* _routing_methods::tx)(call_t *c, packet_t *packet)

Definition at line 272 of file models.h.

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

- models.h

6.30 _shadowing_methods Struct Reference

Methods that should be implemented by a shadowing model.

`#include <models.h>`

Data Fields

- double(* shadowing)(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst, double rxdBm)

6.30.1 Detailed Description

Methods that should be implemented by a shadowing model.

Should use type shadowing_methods_t.

Definition at line 82 of file models.h.

6.30.2 Field Documentation

6.30.2.1 double(* _shadowing_methods::shadowing)(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst, double rxdBm)

Definition at line 83 of file models.h.

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

- models.h

6.31 _worldsens_c_byte_tx Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_node node_id
- ws_id_resource antenna_id
- ws_id_resource wsnet_mod_id
- ws_id_resource wsim_mod_id
- ws_frequency freq
- ws_power power_dbm
- ws_time duration
- ws_time period
- ws_data data

6.31.1 Detailed Description

Definition at line 95 of file worldsens_pkt.h.

6.31.2 Field Documentation

6.31.2.1 ws_id_resource _worldsens_c_byte_tx::antenna_id

Definition at line 98 of file worldsens_pkt.h.

6.31.2.2 ws_data _worldsens_c_byte_tx::data

Definition at line 105 of file worldsens_pkt.h.

6.31.2.3 ws_time _worldsens_c_byte_tx::duration

Definition at line 103 of file worldsens_pkt.h.

6.31.2.4 ws_frequency _worldsens_c_byte_tx::freq

Definition at line 101 of file worldsens_pkt.h.

6.31.2.5 ws_id_node _worldsens_c_byte_tx::node_id

Definition at line 97 of file worldsens_pkt.h.

6.31.2.6 ws_time _worldsens_c_byte_tx::period

Definition at line 104 of file worldsens_pkt.h.

6.31.2.7 ws_power _worldsens_c_byte_tx::power_dbm

Definition at line 102 of file worldsens_pkt.h.

6.31.2.8 ws_pkt_type _worldsens_c_byte_tx::type

Definition at line 96 of file worldsens_pkt.h.

6.31.2.9 ws_id_resource _worldsens_c_byte_tx::wsim_mod_id

Definition at line 100 of file worldsens_pkt.h.

6.31.2.10 ws_id_resource _worldsens_c_byte_tx::wsnet_mod_id

Definition at line 99 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.32 _worldsens_c_connect_req Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_node node_id

6.32.1 Detailed Description

Definition at line 84 of file worldsens_pkt.h.

6.32.2 Field Documentation

6.32.2.1 ws_id_node _worldsens_c_connect_req::node_id

Definition at line 86 of file worldsens_pkt.h.

6.32.2.2 ws_pkt_type _worldsens_c_connect_req::type

Definition at line 85 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.33 _worldsens_c_disconnect Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_node node_id

6.33.1 Detailed Description

Definition at line 115 of file worldsens_pkt.h.

6.33.2 Field Documentation

6.33.2.1 ws_id_node _worldsens_c_disconnect::node_id

Definition at line 117 of file worldsens_pkt.h.

6.33.2.2 ws_pkt_type _worldsens_c_disconnect::type

Definition at line 116 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.34 _worldsens_c_header Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_node id

6.34.1 Detailed Description

Definition at line 79 of file worldsens_pkt.h.

6.34.2 Field Documentation

6.34.2.1 ws_id_node _worldsens_c_header::id

Definition at line 81 of file worldsens_pkt.h.

6.34.2.2 ws_pkt_type _worldsens_c_header::type

Definition at line 80 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.35 _worldsens_c_measure_req Struct Reference

```
#include <worldsens_pkt.h>
```

Data Fields

- ws_pkt_type type
- ws_id_node node_id
- ws_id_resource measure_id
- ws_time period

6.35.1 Detailed Description

Definition at line 108 of file worldsens_pkt.h.

6.35.2 Field Documentation

6.35.2.1 ws_id_resource _worldsens_c_measure_req::measure_id

Definition at line 111 of file worldsens_pkt.h.

6.35.2.2 ws_id_node _worldsens_c_measure_req::node_id

Definition at line 110 of file worldsens_pkt.h.

6.35.2.3 ws_time _worldsens_c_measure_req::period

Definition at line 112 of file worldsens_pkt.h.

6.35.2.4 ws_pkt_type _worldsens_c_measure_req::type

Definition at line 109 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.36 _worldsens_c_sync_ack Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_node node_id
- ws_id_rp rp_id

6.36.1 Detailed Description

Definition at line 89 of file worldsens_pkt.h.

6.36.2 Field Documentation

6.36.2.1 ws_id_node _worldsens_c_sync_ack::node_id

Definition at line 91 of file worldsens_pkt.h.

6.36.2.2 ws_id_rp _worldsens_c_sync_ack::rp_id

Definition at line 92 of file worldsens_pkt.h.

6.36.2.3 ws_pkt_type _worldsens_c_sync_ack::type

Definition at line 90 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.37 _worldsens_pkt Union Reference

`#include <worldsens_pkt.h>`

Data Fields

- struct _worldsens_c_header c_header
- struct _worldsens_s_header s_header
- struct _worldsens_c_connect_req cnx_req
- struct _worldsens_s_connect_rsp_ok cnx_rsp_ok
- struct _worldsens_s_connect_rsp_nok cnx_rsp_nok
- struct _worldsens_c_byte_tx byte_tx
- struct _worldsens_s_byte_rx byte_rx
- struct _worldsens_s_byte_sr_rx byte_sr_rx
- struct _worldsens_c_sync_ack sync_ack
- struct _worldsens_s_sync_release sync_release
- struct _worldsens_s_sync_reminder sync_reminder
- struct _worldsens_s_backtrack bktrk
- struct _worldsens_c_measure_req measure_req
- struct _worldsens_s_measure_rsp measure_rsp
- struct _worldsens_s_measure_sr_rsp measure_sr_rsp
- struct _worldsens_c_disconnect disconnect
- struct _worldsens_s_killsim killsim
- struct _worldsens_s_kill kill

6.37.1 Detailed Description

Definition at line 231 of file worldsens_pkt.h.

6.37.2 Field Documentation

6.37.2.1 struct _worldsens_s_backtrack _worldsens_pkt::bktrk

Definition at line 247 of file worldsens_pkt.h.

6.37.2.2 struct _worldsens_s_byte_rx _worldsens_pkt::byte_rx

Definition at line 240 of file worldsens_pkt.h.

6.37.2.3 struct _worldsens_s_byte_sr_rx _worldsens_pkt::byte_sr_rx

Definition at line 241 of file worldsens_pkt.h.

6.37.2.4 struct _worldsens_c_byte_tx _worldsens_pkt::byte_tx

Definition at line 239 of file worldsens_pkt.h.

6.37.2.5 struct _worldsens_c_header _worldsens_pkt::c_header

Definition at line 232 of file worldsens_pkt.h.

6.37.2.6 struct _worldsens_c_connect_req _worldsens_pkt::cnx_req

Definition at line 235 of file worldsens_pkt.h.

6.37.2.7 struct _worldsens_s_connect_rsp_nok _worldsens_pkt::cnx_rsp_nok

Definition at line 237 of file worldsens_pkt.h.

6.37.2.8 struct _worldsens_s_connect_rsp_ok _worldsens_pkt::cnx_rsp_ok

Definition at line 236 of file worldsens_pkt.h.

6.37.2.9 struct _worldsens_c_disconnect _worldsens_pkt::disconnect

Definition at line 253 of file worldsens_pkt.h.

6.37.2.10 struct _worldsens_s_kill _worldsens_pkt::kill

Definition at line 255 of file worldsens_pkt.h.

6.37.2.11 struct _worldsens_s_killsim _worldsens_pkt::killsim

Definition at line 254 of file worldsens_pkt.h.

6.37.2.12 struct _worldsens_c_measure_req _worldsens_pkt::measure_req

Definition at line 249 of file worldsens_pkt.h.

6.37.2.13 struct _worldsens_s_measure_rsp _worldsens_pkt::measure_rsp

Definition at line 250 of file worldsens_pkt.h.

6.37.2.14 struct _worldsens_s_measure_sr_rsp _worldsens_pkt::measure_sr_rsp

Definition at line 251 of file worldsens_pkt.h.

6.37.2.15 struct _worldsens_s_header _worldsens_pkt::s_header

Definition at line 233 of file worldsens_pkt.h.

6.37.2.16 struct _worldsens_c_sync_ack _worldsens_pkt::sync_ack

Definition at line 243 of file worldsens_pkt.h.

6.37.2.17 struct _worldsens_s_sync_release _worldsens_pkt::sync_release

Definition at line 244 of file worldsens_pkt.h.

6.37.2.18 struct _worldsens_s_sync_reminder _worldsens_pkt::sync_reminder

Definition at line 245 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.38 _worldsens_s_backtrack Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_rp rp_next
- ws_time rp_duration

6.38.1 Detailed Description

Definition at line 159 of file worldsens_pkt.h.

6.38.2 Field Documentation

6.38.2.1 ws_time _worldsens_s_backtrack::rp_duration

Definition at line 163 of file worldsens_pkt.h.

6.38.2.2 ws_id_rp _worldsens_s_backtrack::rp_next

Definition at line 162 of file worldsens_pkt.h.

6.38.2.3 ws_id_seq _worldsens_s_backtrack::seq

Definition at line 161 of file worldsens_pkt.h.

6.38.2.4 ws_pkt_type _worldsens_s_backtrack::type

Definition at line 160 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.39 _worldsens_s_byte_rx Struct Reference

```
#include <worldsens_pkt.h>
```

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_node node_id
- ws_id_resource antenna_id
- ws_id_resource wsim_mod_id
- ws_frequency freq
- ws_power power_dbm
- ws_sinr sinr
- ws_data data

6.39.1 Detailed Description

Definition at line 172 of file worldsens_pkt.h.

6.39.2 Field Documentation

6.39.2.1 ws_id_resource _worldsens_s_byte_rx::antenna_id

Definition at line 176 of file worldsens_pkt.h.

6.39.2.2 ws_data _worldsens_s_byte_rx::data

Definition at line 181 of file worldsens_pkt.h.

6.39.2.3 ws_frequency _worldsens_s_byte_rx::freq

Definition at line 178 of file worldsens_pkt.h.

6.39.2.4 ws_id_node _worldsens_s_byte_rx::node_id

Definition at line 175 of file worldsens_pkt.h.

6.39.2.5 ws_power _worldsens_s_byte_rx::power_dbm

Definition at line 179 of file worldsens_pkt.h.

6.39.2.6 ws_id_seq _worldsens_s_byte_rx::seq

Definition at line 174 of file worldsens_pkt.h.

6.39.2.7 ws_sinr _worldsens_s_byte_rx::sinr

Definition at line 180 of file worldsens_pkt.h.

6.39.2.8 ws_pkt_type _worldsens_s_byte_rx::type

Definition at line 173 of file worldsens_pkt.h.

6.39.2.9 ws_id_resource _worldsens_s_byte_rx::wsim_mod_id

Definition at line 177 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.40 _worldsens_s_byte_sr_rx Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_rp rp_next
- ws_time rp_duration
- ws_id_node node_id
- ws_id_resource antenna_id
- ws_id_resource wsim_mod_id
- ws_frequency freq
- ws_power power_dbm
- ws_sinr sinr
- ws_data data

6.40.1 Detailed Description

Definition at line 184 of file worldsens_pkt.h.

6.40.2 Field Documentation

6.40.2.1 ws_id_resource _worldsens_s_byte_sr_rx::antenna_id

Definition at line 190 of file worldsens_pkt.h.

6.40.2.2 ws_data _worldsens_s_byte_sr_rx::data

Definition at line 195 of file worldsens_pkt.h.

6.40.2.3 ws_frequency _worldsens_s_byte_sr_rx::freq

Definition at line 192 of file worldsens_pkt.h.

6.40.2.4 ws_id_node _worldsens_s_byte_sr_rx::node_id

Definition at line 189 of file worldsens_pkt.h.

6.40.2.5 ws_power _worldsens_s_byte_sr_rx::power_dbm

Definition at line 193 of file worldsens_pkt.h.

6.40.2.6 ws_time _worldsens_s_byte_sr_rx::rp_duration

Definition at line 188 of file worldsens_pkt.h.

6.40.2.7 ws_id_rp _worldsens_s_byte_sr_rx::rp_next

Definition at line 187 of file worldsens_pkt.h.

6.40.2.8 ws_id_seq _worldsens_s_byte_sr_rx::seq

Definition at line 186 of file worldsens_pkt.h.

6.40.2.9 ws_sinr _worldsens_s_byte_sr_rx::sinr

Definition at line 194 of file worldsens_pkt.h.

6.40.2.10 ws_pkt_type _worldsens_s_byte_sr_rx::type

Definition at line 185 of file worldsens_pkt.h.

6.40.2.11 ws_id_resource _worldsens_s_byte_sr_rx::wsim_mod_id

Definition at line 191 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.41 _worldsens_s_connect_rsp_nok Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_seq seq

6.41.1 Detailed Description

Definition at line 147 of file worldsens_pkt.h.

6.41.2 Field Documentation

6.41.2.1 ws_id_seq _worldsens_s_connect_rsp_nok::seq

Definition at line 149 of file worldsens_pkt.h.

6.41.2.2 ws_pkt_type _worldsens_s_connect_rsp_nok::type

Definition at line 148 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.42 _worldsens_s_connect_rsp_ok Struct Reference

```
#include <worldsens_pkt.h>
```

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_rp rp_next
- ws_time rp_duration
- uint8_t n_antenna_id
- uint8_t n_modulation_id
- uint8_t n_measure_id
- char names_and_ids [WORLDSENS_MAX_MODELS_SIZE]

6.42.1 Detailed Description

Definition at line 134 of file worldsens_pkt.h.

6.42.2 Field Documentation

6.42.2.1 uint8_t _worldsens_s_connect_rsp_ok::n_antenna_id

Definition at line 140 of file worldsens_pkt.h.

6.42.2.2 uint8_t _worldsens_s_connect_rsp_ok::n_measure_id

Definition at line 142 of file worldsens_pkt.h.

6.42.2.3 uint8_t _worldsens_s_connect_rsp_ok::n_modulation_id

Definition at line 141 of file worldsens_pkt.h.

6.42.2.4 char _worldsens_s_connect_rsp_ok::names_and_ids[WORLDSENS_MAX_MODELS_SIZE]

Definition at line 144 of file worldsens_pkt.h.

6.42.2.5 ws_time _worldsens_s_connect_rsp_ok::rp_duration

Definition at line 138 of file worldsens_pkt.h.

6.42.2.6 ws_id_rp _worldsens_s_connect_rsp_ok::rp_next

Definition at line 137 of file worldsens_pkt.h.

6.42.2.7 ws_id_seq _worldsens_s_connect_rsp_ok::seq

Definition at line 136 of file worldsens_pkt.h.

6.42.2.8 ws_pkt_type _worldsens_s_connect_rsp_ok::type

Definition at line 135 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.43 _worldsens_s_header Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_seq seq

6.43.1 Detailed Description

Definition at line 129 of file worldsens_pkt.h.

6.43.2 Field Documentation

6.43.2.1 ws_id_seq _worldsens_s_header::seq

Definition at line 131 of file worldsens_pkt.h.

6.43.2.2 ws_pkt_type _worldsens_s_header::type

Definition at line 130 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.44 _worldsens_s_kill Struct Reference

```
#include <worldsens_pkt.h>
```

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_node node_id

6.44.1 Detailed Description

Definition at line 221 of file worldsens_pkt.h.

6.44.2 Field Documentation

6.44.2.1 ws_id_node _worldsens_s_kill::node_id

Definition at line 224 of file worldsens_pkt.h.

6.44.2.2 ws_id_seq _worldsens_s_kill::seq

Definition at line 223 of file worldsens_pkt.h.

6.44.2.3 ws_pkt_type _worldsens_s_kill::type

Definition at line 222 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.45 _worldsens_s_killsim Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_seq seq

6.45.1 Detailed Description

Definition at line 216 of file worldsens_pkt.h.

6.45.2 Field Documentation

6.45.2.1 ws_id_seq _worldsens_s_killsim::seq

Definition at line 218 of file worldsens_pkt.h.

6.45.2.2 ws_pkt_type _worldsens_s_killsim::type

Definition at line 217 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.46 _worldsens_s_measure_rsp Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_node node_id
- ws_id_resource measure_id
- ws_measure measure_val

6.46.1 Detailed Description

Definition at line 198 of file worldsens_pkt.h.

6.46.2 Field Documentation

6.46.2.1 ws_id_resource _worldsens_s_measure_rsp::measure_id

Definition at line 202 of file worldsens_pkt.h.

6.46.2.2 ws_measure _worldsens_s_measure_rsp::measure_val

Definition at line 203 of file worldsens_pkt.h.

6.46.2.3 ws_id_node _worldsens_s_measure_rsp::node_id

Definition at line 201 of file worldsens_pkt.h.

6.46.2.4 ws_id_seq _worldsens_s_measure_rsp::seq

Definition at line 200 of file worldsens_pkt.h.

6.46.2.5 ws_pkt_type _worldsens_s_measure_rsp::type

Definition at line 199 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.47 _worldsens_s_measure_sr_rsp Struct Reference

`#include <worldsens_pkt.h>`

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_node node_id
- ws_id_resource measure_id
- ws_measure measure_val
- ws_id_rp rp_next
- ws_time rp_duration

6.47.1 Detailed Description

Definition at line 206 of file worldsens_pkt.h.

6.47.2 Field Documentation

6.47.2.1 ws_id_resource _worldsens_s_measure_sr_rsp::measure_id

Definition at line 210 of file worldsens_pkt.h.

6.47.2.2 ws_measure _worldsens_s_measure_sr_rsp::measure_val

Definition at line 211 of file worldsens_pkt.h.

6.47.2.3 ws_id_node _worldsens_s_measure_sr_rsp::node_id

Definition at line 209 of file worldsens_pkt.h.

6.47.2.4 ws_time _worldsens_s_measure_sr_rsp::rp_duration

Definition at line 213 of file worldsens_pkt.h.

6.47.2.5 ws_id_rp _worldsens_s_measure_sr_rsp::rp_next

Definition at line 212 of file worldsens_pkt.h.

6.47.2.6 ws_id_seq _worldsens_s_measure_sr_rsp::seq

Definition at line 208 of file worldsens_pkt.h.

6.47.2.7 ws_pkt_type _worldsens_s_measure_sr_rsp::type

Definition at line 207 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.48 _worldsens_s_sync_release Struct Reference

```
#include <worldsens_pkt.h>
```

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_rp rp_next
- ws_time rp_duration

6.48.1 Detailed Description

Definition at line 152 of file worldsens_pkt.h.

6.48.2 Field Documentation

6.48.2.1 ws_time _worldsens_s_sync_release::rp_duration

Definition at line 156 of file worldsens_pkt.h.

6.48.2.2 ws_id_rp _worldsens_s_sync_release::rp_next

Definition at line 155 of file worldsens_pkt.h.

6.48.2.3 ws_id_seq _worldsens_s_sync_release::seq

Definition at line 154 of file worldsens_pkt.h.

6.48.2.4 ws_pkt_type _worldsens_s_sync_release::type

Definition at line 153 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.49 _worldsens_s_sync_reminder Struct Reference

```
#include <worldsens_pkt.h>
```

Data Fields

- ws_pkt_type type
- ws_id_seq seq
- ws_id_rp rp_next

6.49.1 Detailed Description

Definition at line 166 of file worldsens_pkt.h.

6.49.2 Field Documentation

6.49.2.1 ws_id_rp _worldsens_s_sync_reminder::rp_next

Definition at line 169 of file worldsens_pkt.h.

6.49.2.2 ws_id_seq _worldsens_s_sync_reminder::seq

Definition at line 168 of file worldsens_pkt.h.

6.49.2.3 ws_pkt_type _worldsens_s_sync_reminder::type

Definition at line 167 of file worldsens_pkt.h.

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

- worldsens_pkt.h

6.50 bernoulli_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double p

6.50.1 Detailed Description

Definition at line 150 of file probabilistic_distribution.h.

6.50.2 Field Documentation

6.50.2.1 double bernoulli_args_s::p

Definition at line 151 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.51 beta_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double b

6.51.1 Detailed Description

Definition at line 106 of file probabilistic_distribution.h.

6.51.2 Field Documentation

6.51.2.1 double beta_args_s::a

Definition at line 107 of file probabilistic_distribution.h.

6.51.2.2 double beta_args_s::b

Definition at line 108 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.52 binomial_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double p
- unsigned int n

6.52.1 Detailed Description

Definition at line 154 of file probabilistic_distribution.h.

6.52.2 Field Documentation

6.52.2.1 unsigned int binomial_args_s::n

Definition at line 156 of file probabilistic_distribution.h.

6.52.2.2 double binomial_args_s::p

Definition at line 155 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.53 bivariate_gaussian_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double sigma_x
- double sigma_y
- double rho
- double * return_x
- double * return_y

6.53.1 Detailed Description

Definition at line 48 of file probabilistic_distribution.h.

6.53.2 Field Documentation

6.53.2.1 double* bivariate_gaussian_args_s::return_x

Definition at line 52 of file probabilistic_distribution.h.

6.53.2.2 double* bivariate_gaussian_args_s::return_y

Definition at line 53 of file probabilistic_distribution.h.

6.53.2.3 double bivariate_gaussian_args_s::rho

Definition at line 51 of file probabilistic_distribution.h.

6.53.2.4 double bivariate_gaussian_args_s::sigma_x

Definition at line 49 of file probabilistic_distribution.h.

6.53.2.5 double bivariate_gaussian_args_s::sigma_y

Definition at line 50 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.54 cauchy_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a

6.54.1 Detailed Description

Definition at line 69 of file probabilistic_distribution.h.

6.54.2 Field Documentation

6.54.2.1 double cauchy_args_s::a

Definition at line 70 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.55 chisq_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double nu

6.55.1 Detailed Description

Definition at line 102 of file probabilistic_distribution.h.

6.55.2 Field Documentation

6.55.2.1 double chisq_args_s::nu

Definition at line 103 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.56 exponential_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double mu

6.56.1 Detailed Description

Definition at line 56 of file probabilistic_distribution.h.

6.56.2 Field Documentation

6.56.2.1 double exponential_args_s::mu

Definition at line 57 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.57 exponential_s Struct Reference

`#include <timer.h>`

Data Fields

- uint64_t initial_value
- uint64_t ratio
- uint64_t offset
- uint64_t rank

6.57.1 Detailed Description

Definition at line 38 of file timer.h.

6.57.2 Field Documentation

6.57.2.1 uint64_t exponential_s::initial_value

Definition at line 39 of file timer.h.

6.57.2.2 uint64_t exponential_s::offset

Definition at line 41 of file timer.h.

6.57.2.3 uint64_t exponential_s::rank

Definition at line 42 of file timer.h.

6.57.2.4 uint64_t exponential_s::ratio

Definition at line 40 of file timer.h.

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

- timer.h

6.58 exppow_args_s Struct Reference

```
#include <probabilistic_distribution.h>
```

Data Fields

- double a
- double b

6.58.1 Detailed Description

Definition at line 64 of file probabilistic_distribution.h.

6.58.2 Field Documentation

6.58.2.1 double exppow_args_s::a

Definition at line 65 of file probabilistic_distribution.h.

6.58.2.2 double exppow_args_s::b

Definition at line 66 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.59 gamma_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double b

6.59.1 Detailed Description

Definition at line 87 of file probabilistic_distribution.h.

6.59.2 Field Documentation

6.59.2.1 double gamma_args_s::a

Definition at line 88 of file probabilistic_distribution.h.

6.59.2.2 double gamma_args_s::b

Definition at line 89 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.60 gaussian_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double sigma

6.60.1 Detailed Description

Definition at line 39 of file probabilistic_distribution.h.

6.60.2 Field Documentation

6.60.2.1 double gaussian_args_s::sigma

Definition at line 40 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.61 gaussian_tail_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double sigma

6.61.1 Detailed Description

Definition at line 43 of file probabilistic_distribution.h.

6.61.2 Field Documentation

6.61.2.1 double gaussian_tail_args_s::a

Definition at line 44 of file probabilistic_distribution.h.

6.61.2.2 double gaussian_tail_args_s::sigma

Definition at line 45 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.62 geometric_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double p

6.62.1 Detailed Description

Definition at line 159 of file probabilistic_distribution.h.

6.62.2 Field Documentation

6.62.2.1 double geometric_args_s::p

Definition at line 160 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.63 gumbel_t1_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double b

6.63.1 Detailed Description

Definition at line 136 of file probabilistic_distribution.h.

6.63.2 Field Documentation

6.63.2.1 double gumbel_t1_args_s::a

Definition at line 137 of file probabilistic_distribution.h.

6.63.2.2 double gumbel_t1_args_s::b

Definition at line 138 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.64 gumbel_t2_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double b

6.64.1 Detailed Description

Definition at line 141 of file probabilistic_distribution.h.

6.64.2 Field Documentation

6.64.2.1 double gumbel_t2_args_s::a

Definition at line 142 of file probabilistic_distribution.h.

6.64.2.2 double gumbel_t2_args_s::b

Definition at line 143 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.65 hyper_geometric_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- unsigned int n1
- unsigned int n2
- unsigned int t

6.65.1 Detailed Description

Definition at line 163 of file probabilistic_distribution.h.

6.65.2 Field Documentation

6.65.2.1 unsigned int hyper_geometric_args_s::n1

Definition at line 164 of file probabilistic_distribution.h.

6.65.2.2 unsigned int hyper_geometric_args_s::n2

Definition at line 165 of file probabilistic_distribution.h.

6.65.2.3 unsigned int hyper_geometric_args_s::t

Definition at line 166 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.66 laplace_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a

6.66.1 Detailed Description

Definition at line 60 of file probabilistic_distribution.h.

6.66.2 Field Documentation

6.66.2.1 double laplace_args_s::a

Definition at line 61 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.67 levy_alpha_stable_s Struct Reference

```
#include <probabilistic_distribution.h>
```

Data Fields

- double c
- double alpha

6.67.1 Detailed Description

Definition at line 82 of file probabilistic_distribution.h.

6.67.2 Field Documentation

6.67.2.1 double levy_alpha_stable_s::alpha

Definition at line 84 of file probabilistic_distribution.h.

6.67.2.2 double levy_alpha_stable_s::c

Definition at line 83 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.68 logarithmic_args_s Struct Reference

```
#include <probabilistic_distribution.h>
```

Data Fields

- double p

6.68.1 Detailed Description

Definition at line 169 of file probabilistic_distribution.h.

6.68.2 Field Documentation

6.68.2.1 double logarithmic_args_s::p

Definition at line 170 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.69 logistic_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a

6.69.1 Detailed Description

Definition at line 111 of file probabilistic_distribution.h.

6.69.2 Field Documentation

6.69.2.1 double logistic_args_s::a

Definition at line 112 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.70 lognormal_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double zeta
- double sigma

6.70.1 Detailed Description

Definition at line 97 of file probabilistic_distribution.h.

6.70.2 Field Documentation

6.70.2.1 double lognormal_args_s::sigma

Definition at line 99 of file probabilistic_distribution.h.

6.70.2.2 double lognormal_args_s::zeta

Definition at line 98 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.71 pareto_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double b

6.71.1 Detailed Description

Definition at line 115 of file probabilistic_distribution.h.

6.71.2 Field Documentation

6.71.2.1 double pareto_args_s::a

Definition at line 116 of file probabilistic_distribution.h.

6.71.2.2 double pareto_args_s::b

Definition at line 117 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.72 poisson_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double mu

6.72.1 Detailed Description

Definition at line 146 of file probabilistic_distribution.h.

6.72.2 Field Documentation

6.72.2.1 double poisson_args_s::mu

Definition at line 147 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.73 qtimer_s Struct Reference

timer structure parameters: parameters for next_trigger functions (eg: period for periodic timer) conditional_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback_function: function to callback when the timer triggers.

`#include <timer.h>`

Data Fields

- void * trigger_parameters
- int(* conditional_end)(call_t *c, void *timer_id)
- void(* callback_function)(call_t *c, void *timer_id)

- uint64_t(∗ next_trigger)(call_t ∗c, void ∗timer_id)
- call_t ∗ c

6.73.1 Detailed Description

timer structure parameters: parameters for next_trigger functions (eg: period for periodic timer) conditional_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback_function: function to callback when the timer triggers.

Definition at line 18 of file timer.h.

6.73.2 Field Documentation

6.73.2.1 call_t∗ qtimer_s::c

Definition at line 23 of file timer.h.

6.73.2.2 void(∗ qtimer_s::callback_function)(call_t ∗c, void ∗timer_id)

Definition at line 21 of file timer.h.

6.73.2.3 int(∗ qtimer_s::conditional_end)(call_t ∗c, void ∗timer_id)

Definition at line 20 of file timer.h.

6.73.2.4 uint64_t(∗ qtimer_s::next_trigger)(call_t ∗c, void ∗timer_id)

Definition at line 22 of file timer.h.

6.73.2.5 void∗ qtimer_s::trigger_parameters

Definition at line 19 of file timer.h.

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

- timer.h

6.74 rayleigh_s Struct Reference

```
#include <probabilistic_distribution.h>
```

Data Fields

- double sigma

6.74.1 Detailed Description

Definition at line 73 of file probabilistic_distribution.h.

6.74.2 Field Documentation

6.74.2.1 double rayleigh_s::sigma

Definition at line 74 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.75 rayleigh_tail_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double sigma

6.75.1 Detailed Description

Definition at line 77 of file probabilistic_distribution.h.

6.75.2 Field Documentation

6.75.2.1 double rayleigh_tail_s::a

Definition at line 78 of file probabilistic_distribution.h.

6.75.2.2 double rayleigh_tail_s::sigma

Definition at line 79 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.76 spherical_vector_2d_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double * return_x
- double * return_y

6.76.1 Detailed Description

Definition at line 120 of file probabilistic_distribution.h.

6.76.2 Field Documentation

6.76.2.1 double* spherical_vector_2d_args_s::return_x

Definition at line 121 of file probabilistic_distribution.h.

6.76.2.2 double* spherical_vector_2d_args_s::return_y

Definition at line 122 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.77 spherical_vector_3d_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double * return_x
- double * return_y
- double * return_z

6.77.1 Detailed Description

Definition at line 125 of file probabilistic_distribution.h.

6.77.2 Field Documentation

6.77.2.1 double* spherical_vector_3d_args_s::return_x

Definition at line 126 of file probabilistic_distribution.h.

6.77.2.2 double* spherical_vector_3d_args_s::return_y

Definition at line 127 of file probabilistic_distribution.h.

6.77.2.3 double* spherical_vector_3d_args_s::return_z

Definition at line 128 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.78 uniform_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double b

6.78.1 Detailed Description

Definition at line 92 of file probabilistic_distribution.h.

6.78.2 Field Documentation

6.78.2.1 double uniform_args_s::a

Definition at line 93 of file probabilistic_distribution.h.

6.78.2.2 double uniform_args_s::b

Definition at line 94 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

6.79 uniform_random_s Struct Reference

`#include <timer.h>`

Data Fields

- uint64_t min_value
- uint64_t max_value

6.79.1 Detailed Description

Definition at line 45 of file timer.h.

6.79.2 Field Documentation

6.79.2.1 uint64_t uniform_random_s::max_value

Definition at line 47 of file timer.h.

6.79.2.2 uint64_t uniform_random_s::min_value

Definition at line 46 of file timer.h.

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

- timer.h

6.80 weibull_args_s Struct Reference

`#include <probabilistic_distribution.h>`

Data Fields

- double a
- double b

6.80.1 Detailed Description

Definition at line 131 of file probabilistic_distribution.h.

6.80.2 Field Documentation

6.80.2.1 double weibull_args_s::a

Definition at line 132 of file probabilistic_distribution.h.

6.80.2.2 double weibull_args_s::b

Definition at line 133 of file probabilistic_distribution.h.

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

- probabilistic_distribution.h

Chapter 7

File Documentation

7.1 antenna.h File Reference

Antenna declarations.

`#include <include/types.h>`

Functions

- double antenna_get_loss (call_t *c)

  Return the signal loss induced by an antenna circuit.
- angle_t * antenna_get_angle (call_t *c)

  Return the antenna orientation.
- void antenna_set_angle (call_t *c, angle_t *angle)

  Set the antenna orientation.
- void antenna_rx (call_t *c, packet_t *packet)

  Forward a received packet to the antenna.
- double antenna_gain_tx (call_t *c, position_t *position)

  Return the tx antenna gain towards the destination direction.
- double antenna_gain_rx (call_t *c, position_t *position)

  Return the rx antenna gain towards the source direction.

7.1.1 Detailed Description

Antenna declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file antenna.h.

7.1.2 Function Documentation

7.1.2.1 double antenna_gain_rx (call_t * c, position_t * position)

Return the rx antenna gain towards the source direction.

Parameters

c	should be {antenna id, node id, -1}.
position	the source position.

Returns

The antenna gain in dB.

7.1.2.2 double antenna_gain_tx (call_t * c, position_t * position)

Return the tx antenna gain towards the destination direction.

Parameters

c	should be {antenna id, node id, -1}.
position	the destination position.

Returns

The antenna gain in dB.

7.1.2.3 angle_t* antenna_get_angle (call_t * c)

Return the antenna orientation.

Parameters

c	should be {antenna id, node id, -1}.

Returns

Current antenna orientation.

7.1.2.4 double antenna_get_loss (call_t * c)

Return the signal loss induced by an antenna circuit.

Parameters

c	should be {antenna id, node id, -1}.

Returns

Signal loss in dB.

7.1.2.5 void antenna_rx (call_t * c, packet_t * packet)

Forward a received packet to the antenna.

Parameters

c	should be {antenna id, node id, -1}.
packet	the received packet.

7.1.2.6 void antenna_set_angle (call_t * c, angle_t * angle)

Set the antenna orientation.

Parameters

c	should be {antenna id, node id, -1}.
angle	new antenna orientation.

7.2 battery.h File Reference

Battery declarations.

```
#include <include/types.h>
```

Functions

- void battery_consume_tx (call_t *c, uint64_t duration, double txdBm)

  Consume energy associated to a transmission.
- void battery_consume_rx (call_t *c, uint64_t duration)

  Consume energy associated to a reception.
- void battery_consume_idle (call_t *c, uint64_t duration)

  Consume energy associated to idle time.

- void battery_consume (call_t *c, double energy)

  Consume energy.
- double battery_consumed (call_t *c)

  Return the consummed energy.
- double battery_remaining (call_t *c)

  Return the remaining energy.
- double battery_status (call_t *c)

  Returns the percentage of energy charge.

7.2.1 Detailed Description

Battery declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file battery.h.

7.2.2 Function Documentation

7.2.2.1 void battery_consume (call_t * *c*, double *energy*)

Consume energy.

Parameters

c	should be {-1, node id, -1}.
the	amount of consumed energy.

7.2.2.2 void battery_consume_idle (call_t * *c*, uint64_t *duration*)

Consume energy associated to idle time.

Parameters

c	should be {-1, node id, -1}.
duration	the idle duration.

7.2.2.3 void battery_consume_rx (call_t * c, uint64_t duration)

Consume energy associated to a reception.

Parameters

c	should be {-1, node id, -1}.
duration	the reception duration.

7.2.2.4 void battery_consume_tx (call_t * c, uint64_t duration, double txdBm)

Consume energy associated to a transmission.

Parameters

c	should be {-1, node id, -1}.
duration	the transmission duration.
txdBm	the transmission power in dBm.

7.2.2.5 double battery_consumed (call_t * c)

Return the consummed energy.

Parameters

c	should be {-1, node id, -1}.

Returns

Volume of consumed energy.

7.2.2.6 double battery_remaining (call_t * c)

Return the remaining energy.

Parameters

c	should be {-1, node id, -1}.

Returns

Volume of remaining energy.

7.2.2.7 double battery_status (call_t * c)

Returns the percentage of energy charge.

Parameters

c	should be {-1, node id, -1}.

Returns

Percentage of energy charge.

7.3 das.h File Reference

DAta Structure module declarations.

Typedefs

- typedef int(* das_delete_func_t)(void *, void *)

  The prototype of a delete function.

Functions

- int das_init (void)

  Initialize the das module.
- void * das_create (void)

  Create an empty data structure.
- void das_destroy (void *das)

  Destroy a data structure.
- int das_getsize (void *das)

  Return the number of elements in the data structure.
- void das_insert (void *das, void *data)

  Insert an object in the data structure.
- void * das_pop (void *das)

  Remove an arbitrary object from the data structure.
- void * das_pop_FIFO (void *das)

  Remove the oldest object from the data structure.
- int das_find (void *das, void *data)

  Tell if an object belong to the data structure.
- void das_delete (void *das, void *data)

  Remove a particular object from the data structure.
- void das_selective_delete (void *d, das_delete_func_t delete, void *arg)

  Remove objects selected by a delete function from the data structure.

- void das_init_traverse (void *das)

  Initialize a data structure traversal.
- void * das_traverse (void *das)

  Traverse the data structure.

7.3.1 Detailed Description

DAta Structure module declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file das.h.

7.3.2 Typedef Documentation

7.3.2.1 das_delete_func_t

The prototype of a delete function.

Definition at line 14 of file das.h.

7.3.3 Function Documentation

7.3.3.1 void* das_create (void)

Create an empty data structure.

Returns

An opaque pointer to the data structure.

7.3.3.2 void das_delete (void * *das*, void * *data*)

Remove a particular object from the data structure.

Parameters

das	the opaque pointer to the data structure.
data	the object to remove.

7.3.3.3 void das_destroy (void * *das*)

Destroy a data structure.

Objects in the structure are not deallocated.

Parameters

das	the opaque pointer to the data structure.

7.3.3.4 int das_find (void * *das,* void * *data*)

Tell if an object belong to the data structure.

Parameters

das	the opaque pointer to the data structure.
data	the object to find.

Returns

1 if the object is in the data structure, 0 otherwise.

7.3.3.5 int das_getsize (void * *das*)

Return the number of elements in the data structure.

Parameters

das	the opaque pointer to the data structure.

Returns

The number of elements in the data structure.

7.3.3.6 int das_init (void)

Initialize the das module.

Done by the wsnet core.

Returns

0 if success, -1 otherwise.

7.3.3.7 void das_init_traverse (void * das)

Initialize a data structure traversal.

Parameters

das	the opaque pointer to the data structure.

7.3.3.8 void das_insert (void * das, void * data)

Insert an object in the data structure.

Parameters

das	the opaque pointer to the data structure.
data	the object to insert.

7.3.3.9 void* das_pop (void * das)

Remove an arbitrary object from the data structure.

Parameters

das	the opaque pointer to the data structure.

Returns

An arbitrary object.

7.3.3.10 void* das_pop_FIFO (void * das)

Remove the oldest object from the data structure.

Parameters

das	the opaque pointer to the data structure.

Returns

The oldest object.

7.3.3.11 void das_selective_delete (void * d, das_delete_func_t delete, void * arg)

Remove objects selected by a delete function from the data structure.

Parameters

das	the opaque pointer to the data structure.
delete	the function that selects objects to be removed.
arg	an argument passed to the delete function.

7.3.3.12 void* das_traverse (void * *das*)

Traverse the data structure.

Parameters

das	the opaque pointer to the data structure.

Returns

The next object in the traversal.

7.4 dbg.h File Reference

dbg declarations

Defines

- #define DEBUG_MAX 9
- #define DBG(lvl, msg,...)
- #define NDBG(lvl, msg, c,...)
- #define DBG_CRIT(msg,...) DBG(0, msg, ## __VA_ARGS__)
- #define NDBG_CRIT(msg, c,...) NDBG(0, msg, c, ## __VA_ARGS__)
- #define DBG_WARN(msg,...) DBG(1, msg, ## __VA_ARGS__)
- #define NDBG_WARN(msg,...) NDBG(1, msg, c, ## __VA_ARGS__)
- #define DBG_INFO(msg,...) DBG(2, msg, ## __VA_ARGS__)
- #define NDBG_INFO(msg, c,...) NDBG(2, msg, c, ## __VA_ARGS__)
- #define DBG_VERB(msg,...) DBG(3, msg, ## __VA_ARGS__)
- #define NDBG_VERB(msg, c,...) NDBG(3, msg, c, ## __VA_ARGS__)
- #define DBG_NOISE(msg,...) DBG(4, msg, ## __VA_ARGS__)
- #define NDBG_NOISE(msg, c,...) NDBG(4, msg, c, ## __VA_ARGS__)
- #define DBG_XTRM(msg,...) DBG(5, msg, ## __VA_ARGS__)
- #define NDBG_XTRM(msg, c,...) NDBG(5, msg, c, ## __VA_ARGS__)

Functions

- static void get_debug (call_t *c, char **entity_name, int *debug, int *log)

7.4.1 Detailed Description

dbg declarations

Author

Stephane D'Alu

Date

2008

Definition in file dbg.h.

7.4.2 Define Documentation

7.4.2.1 #define DBG(*lvl, msg, ...*)

Value:

```
if (lvl <= DBG_VAR) {                                           \
        fprintf(stderr, "%s: ", DBG_NAME);                              \
        fprintf(stderr, msg "\n", ##  __VA_ARGS__);                     \
    }
```

Definition at line 16 of file dbg.h.

7.4.2.2 #define DBG_CRIT(*msg, ...*) DBG(0, msg, ## __VA_ARGS__)

Definition at line 32 of file dbg.h.

7.4.2.3 #define DBG_INFO(*msg, ...*) DBG(2, msg, ## __VA_ARGS__)

Definition at line 48 of file dbg.h.

7.4.2.4 #define DBG_NOISE(*msg, ...*) DBG(4, msg, ## __VA_ARGS__)

Definition at line 64 of file dbg.h.

7.4.2.5 #define DBG_VERB(*msg, ...*) DBG(3, msg, ## __VA_ARGS__)

Definition at line 56 of file dbg.h.

7.4.2.6 #define DBG_WARN(*msg, ...*) DBG(1, msg, ## __VA_ARGS__)

Definition at line 40 of file dbg.h.

7.4.2.7 #define DBG_XTRM(*msg, ...*) DBG(5, msg, ## __VA_ARGS__)

Definition at line 72 of file dbg.h.

7.4.2.8 #define DEBUG_MAX 9

Definition at line 12 of file dbg.h.

7.4.2.9 #define NDBG(*lvl, msg, c, ...*)

Value:

```
if (lvl <= DBG_VAR) {                                           \
        fprintf(stderr, "%s[%3d]: ", DBG_NAME, (c)->node);      \
        fprintf(stderr, msg "\n", ##  __VA_ARGS__);              \
    }
```

Definition at line 21 of file dbg.h.

7.4.2.10 #define NDBG_CRIT(*msg, c, ...*) NDBG(0, msg, c, ## __VA_ARGS__)

Definition at line 33 of file dbg.h.

7.4.2.11 #define NDBG_INFO(*msg, c, ...*) NDBG(2, msg, c, ## __VA_ARGS__)

Definition at line 49 of file dbg.h.

7.4.2.12 #define NDBG_NOISE(*msg, c, ...*) NDBG(4, msg, c, ## __VA_ARGS__)

Definition at line 65 of file dbg.h.

7.4.2.13 #define NDBG_VERB(*msg, c, ...*) NDBG(3, msg, c, ## __VA_ARGS__)

Definition at line 57 of file dbg.h.

7.4.2.14 #define NDBG_WARN(*msg, ...*) NDBG(1, msg, c, ## __VA_ARGS__)

Definition at line 41 of file dbg.h.

7.4.2.15 #define NDBG_XTRM(*msg, c, ...*) NDBG(5, msg, c, ## __VA_ARGS__)

Definition at line 73 of file dbg.h.

7.4.3 Function Documentation

7.4.3.1 static void get_debug (call_t * c, char ** entity_name, int * debug, int * log) [inline, static]

Definition at line 81 of file dbg.h.

References get_entity_name().

7.5 entity.h File Reference

Entity declarations.

```
#include <include/types.h>
```

Functions

- char * get_entity_name (call_t *c)
  Return the name of an entity.
- void * get_entity_private_data (call_t *c)
  Return the private data associated to an entity.
- void set_entity_private_data (call_t *c, void *data)
  Associate a private data memory space with an entity.
- void * get_node_private_data (call_t *c)
  Return the private data associated to an (entity, node).
- void set_node_private_data (call_t *c, void *data)
  Associate a private data memory space with an (entity, node).
- int get_entity_type (call_t *c)
  Return the type of an entity.
- array_t * get_entity_bindings_up (call_t *c)
  Return an array containing the entities that are up "c->entity" in "c->node".
- array_t * get_entity_bindings_down (call_t *c)
  Return an array containing the entities that are down "c->entity" in "c->node".
- array_t * get_application_entities (call_t *c)
  Return an array containing the application entities in "c->node".
- array_t * get_routing_entities (call_t *c)
  Return an array containing the routing entities in "c->node".
- array_t * get_mac_entities (call_t *c)
  Return an array containing the mac entities in "c->node".
- array_t * get_radio_entities (call_t *c)
  Return an array containing the radio entities in "c->node".
- array_t * get_antenna_entities (call_t *c)
  Return an array containing the antenna entities in "c->node".
- entityid_t get_energy_entity (call_t *c)

Return an array containing the energy entity in "c->node".

- entityid_t get_mobility_entity (call_t *c)

  Return an array containing the mobility entity in "c->node".

- int get_entity_links_up_nbr (call_t *c)

  Return the number of entities that are linked up with "c->entity" in "c->node".

- entityid_t * get_entity_links_up (call_t *c)

  Return an array containing the entity ids of the entities linked up with "c->entity" in "c->node".

- int get_entity_links_down_nbr (call_t *c)

  Return the number of entities that are linked down with "c->entity" in "c->node".

- entityid_t * get_entity_links_down (call_t *c)

  Return an array containing the entity ids of the entities linked down with "c->entity" in "c->node".

7.5.1 Detailed Description

Entity declarations. Parameter parsing declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file entity.h.

7.5.2 Function Documentation

7.5.2.1 array_t* get_antenna_entities (call_t * *c*)

Return an array containing the antenna entities in "c->node".

Parameters

c	sould be {-1, node id, -1}.

Returns

An array of entity ids.

7.5.2.2 array_t* get_application_entities (call_t * *c*)

Return an array containing the application entities in "c->node".

Parameters

c	sould be {-1, node id, -1}.

Returns

An array of entity ids.

7.5.2.3 entityid_t get_energy_entity (call_t * c)

Return an array containing the energy entity in "c->node".

Parameters

c	sould be {-1, node id, -1}.

Returns

An entity id.

7.5.2.4 array_t* get_entity_bindings_down (call_t * c)

Return an array containing the entities that are down "c->entity" in "c->node".

Parameters

c	sould be {entity id, node id, -1}.

Returns

An array of entity ids.

7.5.2.5 array_t* get_entity_bindings_up (call_t * c)

Return an array containing the entities that are up "c->entity" in "c->node".

Parameters

c	sould be {entity id, node id, -1}.

Returns

An array of entity ids.

7.5.2.6 entityid_t* get_entity_links_down (call_t * c)

Return an array containing the entity ids of the entities linked down with "c->entity" in "c->node".

Parameters

c	sould be {entity id, node id, -1}.

Returns

Array of entity ids.

Deprecated Should use get_entity_bindings_down() instead.

7.5.2.7 int get_entity_links_down_nbr (call_t * c)

Return the number of entities that are linked down with "c->entity" in "c->node".

Parameters

c	sould be {entity id, node id, -1}.

Returns

The number of linked down entities.

Deprecated Should use get_entity_bindings_down() instead.

7.5.2.8 entityid_t* get_entity_links_up (call_t * c)

Return an array containing the entity ids of the entities linked up with "c->entity" in "c->node".

Parameters

c	sould be {entity id, node id, -1}.

Returns

Array of entity ids.

Deprecated Should use get_entity_bindings_up() instead.

7.5.2.9 int get_entity_links_up_nbr (call_t * c)

Return the number of entities that are linked up with "c->entity" in "c->node".

Parameters

c	sould be {entity id, node id, -1}.

Returns

The number of linked up entities.

Deprecated Should use get_entity_bindings_up() instead.

7.5.2.10 char* get_entity_name (call_t * c)

Return the name of an entity.

Parameters

c	sould be {entity id, -1, -1}.

Returns

The name of the entity.

Referenced by get_debug().

7.5.2.11 void* get_entity_private_data (call_t * c)

Return the private data associated to an entity.

Parameters

c	sould be {entity id, -1, -1}.

Returns

A (void *) pointer to the entity private data.

7.5.2.12 int get_entity_type (call_t * c)

Return the type of an entity.

Parameters

c	sould be {entity id, -1, -1}.

Returns

The entity type.

7.5.2.13 array_t* get_mac_entities (call_t * c)

Return an array containing the mac entities in "c->node".

Parameters

c	sould be {-1, node id, -1}.

Returns

An array of entity ids.

7.5.2.14 entityid_t get_mobility_entity (call_t * c)

Return an array containing the mobility entity in "c->node".

Parameters

c	sould be {-1, node id, -1}.

Returns

An entity id.

7.5.2.15 void* get_node_private_data (call_t * c)

Return the private data associated to an (entity, node).

Parameters

c	sould be {entity id, node id, -1}.

Returns

A (void *) pointer to the (entity, node) private data.

7.5.2.16 array_t* get_radio_entities (call_t * c)

Return an array containing the radio entities in "c->node".

Parameters

c	sould be {-1, node id, -1}.

Returns

An array of entity ids.

7.5.2.17 array_t* get_routing_entities (call_t * c)

Return an array containing the routing entities in "c->node".

Parameters

c	sould be {-1, node id, -1}.

Returns

An array of entity ids.

7.5.2.18 void set_entity_private_data (call_t * c, void * data)

Associate a private data memory space with an entity.

Parameters

c	sould be {entity id, -1, -1}.
data	a (void *) pointer to the private memory space.

7.5.2.19 void set_node_private_data (call_t * c, void * data)

Associate a private data memory space with an (entity, node).

Parameters

c	sould be {entity id, node id, -1}.
data	a (void *) pointer to the private memory space.

7.6 hadas.h File Reference

Hashed DAta Structure module declarations.

Typedefs

- typedef unsigned long(∗ hash_hash_t)(void ∗key)
  The prototype of a hash function.
- typedef int(∗ hash_equal_t)(void ∗key0, void ∗key1)
  The prototype of an equal function.

Functions

- int hadas_init (void)
  Initialize the hadas module.
- void ∗ hadas_create (hash_hash_t hash, hash_equal_t equal)
  Create an empty hadas structure.
- void hadas_destroy (void ∗hadas)
  Destroy a hadas structure.
- void hadas_insert (void ∗hadas, void ∗key, void ∗data)
  Insert a (key,object) in the hadas structure.
- void ∗ hadas_get (void ∗hadas, void ∗key)
  Return the object associated to a key in the hadas structure.
- void hadas_delete (void ∗hadas, void ∗key)
  Remove the object associated to a key in the hadas structure.

7.6.1 Detailed Description

Hashed DAta Structure module declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file hadas.h.

7.6.2 Typedef Documentation

7.6.2.1 hash_equal_t

The prototype of an equal function.

Definition at line 20 of file hadas.h.

7.6.2.2 hash_hash_t

The prototype of a hash function.

Definition at line 14 of file hadas.h.

7.6.3 Function Documentation

7.6.3.1 void* hadas_create (hash_hash_t *hash*, hash_equal_t *equal*)

Create an empty hadas structure.

Parameters

hash	the function used to hash object keys.
equal	the function used to decide equality between keys.

Returns

An opaque pointer to the hadas structure.

7.6.3.2 void hadas_delete (void * *hadas*, void * *key*)

Remove the object associated to a key in the hadas structure.

Parameters

hadas	the opaque pointer to the data structure.
key	the key of the object to be removed.

7.6.3.3 void hadas_destroy (void * *hadas*)

Destroy a hadas structure.

Objects and keys in the structure are not deallocated.

Parameters

hadas	the opaque pointer to the hadas structure.

7.6.3.4 void* hadas_get (void * *hadas*, void * *key*)

Return the object associated to a key in the hadas structure.

Parameters

hadas	the opaque pointer to the data structure.
key	the key of the object to get.

Returns

The object.

7.6.3.5 int hadas_init (void)

Initialize the hadas module.

Done by the wsnet core.

Returns

0 if success, -1 otherwise.

7.6.3.6 void hadas_insert (void * *hadas*, void * *key*, void * *data*)

Insert a (key,object) in the hadas structure.

Parameters

hadas	the opaque pointer to the data structure.
key	the key of the object to insert.
data	the object to insert.

7.7 ioctl_message.h File Reference

Functions

- ioctl_message_t * ioctl_message_create (int type, int size, int real_size, void *body)

  Create an ioctl message.
- void ioctl_message_dealloc (void *message)

  Deallocate an ioctl message.
- void ioctl_message_body_duplicate (void *message, void *pointer)

  Duplicate an ioctl message.
- void * get_ioctl_message_body (void *message)

  Return the message body.
- int get_ioctl_message_type (void *message)

  Return the message type.
- int get_ioctl_message_size (void *message)

Return the size of the message body.

- int get_ioctl_message_real_size (void *message)

  Return the real size of the message body.

7.7.1 Function Documentation

7.7.1.1 void* get_ioctl_message_body (void * *message*)

Return the message body.

Parameters

message	the message.

7.7.1.2 int get_ioctl_message_real_size (void * *message*)

Return the real size of the message body.

Parameters

message	the message.

7.7.1.3 int get_ioctl_message_size (void * *message*)

Return the size of the message body.

Parameters

message	the message.

7.7.1.4 int get_ioctl_message_type (void * *message*)

Return the message type.

Parameters

message	the message.

7.7.1.5 void ioctl_message_body_duplicate (void * *message,* void * *pointer*)

Duplicate an ioctl message.

Parameters

message	the message to duplicate.

7.7.1.6 ioctl_message_t* ioctl_message_create (int *type*, int *size*, int *real_size*, void * *body*)

Create an ioctl message.

Parameters

type	type of the message.
size	size of the message.
real_size	real size of the message.
body	body of the message.

7.7.1.7 void ioctl_message_dealloc (void * *message*)

Deallocate an ioctl message.

Parameters

message	the message to deallocate.

7.8 log.h File Reference

log declarations

```
#include <include/options.h>
```

Defines

- #define PRINT_REPLAY(x...) do { } while (0)
- #define PRINT_APPLICATION(x...) do { } while (0)
- #define PRINT_ROUTING(x...) do { } while (0)
- #define PRINT_MAC(x...) do { } while (0)
- #define PRINT_RADIO(x...) do { } while (0)
- #define PRINT_ANTENNA(x...) do { } while (0)
- #define PRINT_MOBILITY(x...) do { } while (0)
- #define PRINT_ENERGY(x...) do { } while (0)
- #define PRINT_ENVIRONMENT(x...) do { } while (0)
- #define PRINT_MONITOR(x...) do { } while (0)
- #define PRINT_MODULATION(x...) do { } while (0)
- #define PRINT_INTERFERENCES(x...) do { } while (0)
- #define PRINT_PROPAGATION(x...) do { } while (0)
- #define PRINT_WORLDSENS(x...) do { } while (0)

7.8.1 Detailed Description

log declarations

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file log.h.

7.8.2 Define Documentation

7.8.2.1 #define PRINT_ANTENNA(*x...*) do { } while (0)

Definition at line 45 of file log.h.

7.8.2.2 #define PRINT_APPLICATION(*x...*) do { } while (0)

Definition at line 21 of file log.h.

7.8.2.3 #define PRINT_ENERGY(*x...*) do { } while (0)

Definition at line 57 of file log.h.

7.8.2.4 #define PRINT_ENVIRONMENT(*x...*) do { } while (0)

Definition at line 63 of file log.h.

7.8.2.5 #define PRINT_INTERFERENCES(*x...*) do { } while (0)

Definition at line 81 of file log.h.

7.8.2.6 #define PRINT_MAC(*x...*) do { } while (0)

Definition at line 33 of file log.h.

7.8.2.7 #define PRINT_MOBILITY(*x...*) do { } while (0)

Definition at line 51 of file log.h.

7.8.2.8 #define PRINT_MODULATION(x...) do { } while (0)

Definition at line 75 of file log.h.

7.8.2.9 #define PRINT_MONITOR(x...) do { } while (0)

Definition at line 69 of file log.h.

7.8.2.10 #define PRINT_PROPAGATION(x...) do { } while (0)

Definition at line 87 of file log.h.

7.8.2.11 #define PRINT_RADIO(x...) do { } while (0)

Definition at line 39 of file log.h.

7.8.2.12 #define PRINT_REPLAY(x...) do { } while (0)

Definition at line 15 of file log.h.

7.8.2.13 #define PRINT_ROUTING(x...) do { } while (0)

Definition at line 27 of file log.h.

7.8.2.14 #define PRINT_WORLDSENS(x...) do { } while (0)

Definition at line 94 of file log.h.

7.9 measure.h File Reference

Measure declarations.

```
#include <include/types.h>
```

Functions

- measureid_t get_measureid_by_name (char *name)

  Return the measure id associated to a measure name.
- void READ_MEASURE (call_t *c, measureid_t measure, double *value)

  Return the measure id associated to a measure name.

7.9.1 Detailed Description

Measure declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file measure.h.

7.9.2 Function Documentation

7.9.2.1 measureid_t get_measureid_by_name (char * *name*)

Return the measure id associated to a measure name.

Parameters

name	the measure name.

Returns

The measure id.

7.9.2.2 void READ_MEASURE (call_t * *c*, measureid_t *measure*, double * *value*)

Return the measure id associated to a measure name.

Parameters

c	should be {caller entity id, caller node id, -1}.
measure	the measure id.
value	a pointer where to put the read value.

7.10 medium.h File Reference

Medium declarations.

`#include <include/types.h>` `#include <math.h>`

Defines

- #define MIN_DBM -DBL_MAX

  Signal strength of a null signal in dBm.
- #define MAX_SNR DBL_MAX

  Maximum SNR value, when there is no interference nor noise.

Functions

- static double dBm2mW (double dBm)

  Convert a dBm value into a mW value.
- static double mW2dBm (double mW)

  Convert a mW value into a dBm value.
- void MEDIA_TX (call_t *c, packet_t *packet)

  Transmit a packet in the radio medium.
- double MEDIA_GET_NOISE (call_t *c, int channel)

  Return the radio medium noise on a given channel.

7.10.1 Detailed Description

Medium declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file medium.h.

7.10.2 Define Documentation

7.10.2.1 #define MAX_SNR DBL_MAX

Maximum SNR value, when there is no interference nor noise.

Definition at line 26 of file medium.h.

7.10.2.2 #define MIN_DBM -DBL_MAX

Signal strength of a null signal in dBm.

Definition at line 19 of file medium.h.

Referenced by dBm2mW(), and mW2dBm().

7.10.3 Function Documentation

7.10.3.1 static double dBm2mW (double *dBm*) [inline, static]

Convert a dBm value into a mW value.

Parameters

dBm	the dBm value to convert.

Returns

The converted mW value.

Definition at line 34 of file medium.h.

References MIN_DBM.

7.10.3.2 double MEDIA_GET_NOISE (call_t * *c,* int *channel*)

Return the radio medium noise on a given channel.

Parameters

c	should be {radio id, node id, antenna id}.
channel	the radio channel we are listening to.

Returns

the noise strength in dBm.

7.10.3.3 void MEDIA_TX (call_t * *c,* packet_t * *packet*)

Transmit a packet in the radio medium.

Parameters

c	should be {radio id, node id, antenna id}.
packet	the transmitted packet.

7.10.3.4 static double mW2dBm (double *mW*) [inline, static]

Convert a mW value into a dBm value.

Parameters

mW	the mW value to convert.

Returns

The converted dBm value.

Definition at line 44 of file medium.h.

References MIN_DBM.

7.11 mem_fs.h File Reference

Fixed size memory management module declarations.

Functions

- void mem_fs_clean (void)
  Clean the mem_fs module.
- void * mem_fs_slice_declare (int size)
  Declare a memory allocation slice.
- void * mem_fs_alloc (void *slice)
  Allocate a memory block from a slice.
- void mem_fs_dealloc (void *slice, void *pointer)
  Deallocate a memory block.

7.11.1 Detailed Description

Fixed size memory management module declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file mem_fs.h.

7.11.2 Function Documentation

7.11.2.1 void* mem_fs_alloc (void * *slice*)

Allocate a memory block from a slice.

Parameters

slice	the opaque pointer to the memory slice.

Returns

The allocated memory block.

7.11.2.2 void mem_fs_clean (void)

Clean the mem_fs module.

Done by the wsnet core.

7.11.2.3 void mem_fs_dealloc (void * *slice*, void * *pointer*)

Deallocate a memory block.

Parameters

slice	the opaque pointer to the memory slice.

Returns

pointer the memory block to deallocate.

7.11.2.4 void* mem_fs_slice_declare (int *size*)

Declare a memory allocation slice.

Parameters

size	the size of the memory blcoks that will be allocated using this slice.

Returns

An opaque pointer to the memory slice.

7.12 models.h File Reference

Models declarations.

```
#include <include/types.h>
```

Data Structures

- struct _model

  Information about a model.
- struct _propagation_methods

Methods that should be implemented by a propagation model.

- struct _fading_methods

  Methods that should be implemented by a fading model.
- struct _shadowing_methods

  Methods that should be implemented by a shadowing model.
- struct _interferences_methods

  Methods that should be implemented by a interferences model.
- struct _noise_methods

  Methods that should be implemented by a noise model.
- struct _modulation_methods

  Methods that should be implemented by a modulation model.
- struct _environment_methods

  Methods that should be implemented by an environment model.
- struct _monitor_methods

  Methods that should be implemented by a monitor model.
- struct _mobility_methods

  Methods that should be implemented by a mobility model.
- struct _energy_methods

  Methods that should be implemented by an energy model.
- struct _antenna_methods

  Methods that should be implemented by an antenna model.
- struct _radio_methods

  Methods that should be implemented by a radio model.
- struct _mac_methods

  Methods that should be implemented by a mac model.
- struct _routing_methods

  Methods that should be implemented by a routing model.
- struct _application_methods

  Methods that should be implemented by an application model.

Defines

- #define MODELTYPE_PROPAGATION 0
- #define MODELTYPE_SHADOWING 1
- #define MODELTYPE_FADING 2
- #define MODELTYPE_INTERFERENCES 3
- #define MODELTYPE_NOISE 4
- #define MODELTYPE_MODULATION 5
- #define MODELTYPE_ENVIRONMENT 6
- #define MODELTYPE_MOBILITY 7
- #define MODELTYPE_ANTENNA 8
- #define MODELTYPE_RADIO 9
- #define MODELTYPE_MAC 10
- #define MODELTYPE_ROUTING 11

- #define MODELTYPE_APPLICATION 12
- #define MODELTYPE_ENERGY 13
- #define MODELTYPE_MONITOR 14

Typedefs

- typedef struct _model model_t

  Information about a model.
- typedef struct _propagation_methods propagation_methods_t

  Methods that should be implemented by a propagation model.
- typedef struct _fading_methods fading_methods_t

  Methods that should be implemented by a fading model.
- typedef struct _shadowing_methods shadowing_methods_t

  Methods that should be implemented by a shadowing model.
- typedef struct _interferences_methods interferences_methods_t

  Methods that should be implemented by a interferences model.
- typedef struct _noise_methods noise_methods_t

  Methods that should be implemented by a noise model.
- typedef struct _modulation_methods modulation_methods_t

  Methods that should be implemented by a modulation model.
- typedef struct _environment_methods environment_methods_t

  Methods that should be implemented by an environment model.
- typedef struct _monitor_methods monitor_methods_t

  Methods that should be implemented by a monitor model.
- typedef struct _mobility_methods mobility_methods_t

  Methods that should be implemented by a mobility model.
- typedef struct _energy_methods energy_methods_t

  Methods that should be implemented by an energy model.
- typedef struct _antenna_methods antenna_methods_t

  Methods that should be implemented by an antenna model.
- typedef struct _radio_methods radio_methods_t

  Methods that should be implemented by a radio model.
- typedef struct _mac_methods mac_methods_t

  Methods that should be implemented by a mac model.
- typedef struct _routing_methods routing_methods_t

  Methods that should be implemented by a routing model.
- typedef struct _application_methods application_methods_t

  Methods that should be implemented by an application model.

7.12.1 Detailed Description

Models declarations.

Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

Date

2007

Definition in file models.h.

7.12.2 Define Documentation

7.12.2.1 #define MODELTYPE_ANTENNA 8

Definition at line 21 of file models.h.

7.12.2.2 #define MODELTYPE_APPLICATION 12

Definition at line 25 of file models.h.

7.12.2.3 #define MODELTYPE_ENERGY 13

Definition at line 26 of file models.h.

7.12.2.4 #define MODELTYPE_ENVIRONMENT 6

Definition at line 19 of file models.h.

7.12.2.5 #define MODELTYPE_FADING 2

Definition at line 15 of file models.h.

7.12.2.6 #define MODELTYPE_INTERFERENCES 3

Definition at line 16 of file models.h.

7.12.2.7 #define MODELTYPE_MAC 10

Definition at line 23 of file models.h.

7.12.2.8 #define MODELTYPE_MOBILITY 7

Definition at line 20 of file models.h.

7.12.2.9 #define MODELTYPE_MODULATION 5

Definition at line 18 of file models.h.

7.12.2.10 #define MODELTYPE_MONITOR 14

Definition at line 27 of file models.h.

7.12.2.11 #define MODELTYPE_NOISE 4

Definition at line 17 of file models.h.

7.12.2.12 #define MODELTYPE_PROPAGATION 0

Definition at line 13 of file models.h.

7.12.2.13 #define MODELTYPE_RADIO 9

Definition at line 22 of file models.h.

7.12.2.14 #define MODELTYPE_ROUTING 11

Definition at line 24 of file models.h.

7.12.2.15 #define MODELTYPE_SHADOWING 1

Definition at line 14 of file models.h.

7.12.3 Typedef Documentation

7.12.3.1 antenna_methods_t

Methods that should be implemented by an antenna model.

7.12.3.2 application_methods_t

Methods that should be implemented by an application model.

7.12.3.3 energy_methods_t

Methods that should be implemented by an energy model.

7.12.3.4 environment_methods_t

Methods that should be implemented by an environment model.

7.12.3.5 fading_methods_t

Methods that should be implemented by a fading model.

7.12.3.6 interferences_methods_t

Methods that should be implemented by a interferences model.

7.12.3.7 mac_methods_t

Methods that should be implemented by a mac model.

7.12.3.8 mobility_methods_t

Methods that should be implemented by a mobility model.

7.12.3.9 model_t

Information about a model.

7.12.3.10 modulation_methods_t

Methods that should be implemented by a modulation model.

7.12.3.11 monitor_methods_t

Methods that should be implemented by a monitor model.

7.12.3.12 noise_methods_t

Methods that should be implemented by a noise model.

7.12.3.13 propagation_methods_t

Methods that should be implemented by a propagation model.

7.12.3.14 radio_methods_t

Methods that should be implemented by a radio model.

7.12.3.15 routing_methods_t

Methods that should be implemented by a routing model.

7.12.3.16 shadowing_methods_t

Methods that should be implemented by a shadowing model.

7.13 modelutils.h File Reference

Utility function declarations.

```
#include <stdlib.h> #include <stdio.h> #include <string.-
h> #include <float.h> #include <limits.h> #include <include/options.-
h>  #include <include/log.h>  #include <include/mem_fs.-
h> #include <include/das.h> #include <include/sodas.h> ×
#include <include/spadas.h>   #include <include/hadas.h>
#include <include/timer.h>  #include <include/types.h> ×
#include <include/models.h>   #include <include/rng.h> ×
#include <include/probabilistic_distribution.h>  #include
<include/medium.h>  #include <include/radio.h>  #include
<include/antenna.h> #include <include/battery.h> #include
<include/entity.h>  #include <include/packet.h>  #include
<include/node.h>   #include <include/param.h>   #include
<include/measure.h>     #include <include/scheduler.h> ×
#include <include/monitor.h>   #include <include/ioctl_-
message.h> #include <include/modulation.h>
```

Defines

- #define BROADCAST_ADDR -1

  Broadcast address.

Functions

- void end_simulation (void)

Stop the simulation.

- uint64_t get_time (void)

  Return the current simulation time.

- int get_node_count (void)

  Return the number of simulated nodes.

- position_t * get_topology_area (void)

  Return the size of the network area.

- void TX (call_t *c, packet_t *packet)
- void RX (call_t *c, packet_t *packet)
- int IOCTL (call_t *c, int option, void *in, void **out)
- int SET_HEADER (call_t *c, packet_t *packet, destination_t *dst)
- int GET_HEADER_SIZE (call_t *c)
- int GET_HEADER_REAL_SIZE (call_t *c)

7.13.1 Detailed Description

Utility function declarations.

Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

Date

2007

Definition in file modelutils.h.

7.13.2 Define Documentation

7.13.2.1 #define BROADCAST_ADDR -1

Broadcast address.

Definition at line 45 of file modelutils.h.

7.13.3 Function Documentation

7.13.3.1 void end_simulation (void)

Stop the simulation.

7.13.3.2 int GET_HEADER_REAL_SIZE (call_t * c)

7.13.3.3 int GET_HEADER_SIZE (call_t * c)

7.13.3.4 int get_node_count (void)

Return the number of simulated nodes.

Returns

The number of ndoes.

7.13.3.5 uint64_t get_time (void)

Return the current simulation time.

Returns

The simulation time.

7.13.3.6 position_t* get_topology_area (void)

Return the size of the network area.

Returns

The network area.

7.13.3.7 int IOCTL (call_t * c, int option, void * in, void ** out)

7.13.3.8 void RX (call_t * c, packet_t * packet)

7.13.3.9 int SET_HEADER (call_t * c, packet_t * packet, destination_t * dst)

7.13.3.10 void TX (call_t * c, packet_t * packet)

7.14 modulation.h File Reference

Modulation declarations.

```
#include <include/types.h>
```

Functions

- int modulation_bit_per_symbol (entityid_t modulation)

  Returns the bit per sumbol rate for a given modulation scheme.

7.14.1 Detailed Description

Modulation declarations.

Author

Guillaume Chelius & Elyes Ben Hamida & Christophe Savigny

Date

2009

Definition in file modulation.h.

7.14.2 Function Documentation

7.14.2.1 int modulation_bit_per_symbol (entityid_t *modulation*)

Returns the bit per sumbol rate for a given modulation scheme.

Parameters

modulation	: modulation id.

Returns

The bit per symbol.

7.15 monitor.h File Reference

Monitor declarations.

```
#include <include/types.h>
```

Functions

- void monitor_simulation (void)

  Call the "monitor_event()" function of the monitoring entity.
- void monitor_register_callback (callback_t callback, call_t *c, void *arg)

  Register a function that is called whenever the network is monitored.

7.15.1 Detailed Description

Monitor declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file monitor.h.

7.15.2 Function Documentation

7.15.2.1 void monitor_register_callback (callback_t *callback*, call_t * *c*, void * *arg*)

Register a function that is called whenever the network is monitored.

Parameters

callback	the function that is called back.
c	the call parameter given to the callback function.
arg	a paramater that given to the callback function.

7.15.2.2 void monitor_simulation (void)

Call the "monitor_event()" function of the monitoring entity.

7.16 node.h File Reference

Node declarations.

```
#include <include/types.h>
```

Functions

- double distance (position_t *position0, position_t *position1)

  Return the distance between two points.
- position_t * get_node_position (nodeid_t node)

  Return a node's position.
- void node_kill (nodeid_t id)

  Kill a node during the simulation.
- int is_node_alive (nodeid_t id)

  Check wether a node is alive.

7.16.1 Detailed Description

Node declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file node.h.

7.16.2 Function Documentation

7.16.2.1 double distance (position_t * position0, position_t * position1)

Return the distance between two points.

Parameters

position0	first point.
position1	second point.

Returns

The distance between point0 and point1.

7.16.2.2 position_t* get_node_position (nodeid_t node)

Return a node's position.

For read-only, the position should not be modified.

Parameters

node	the node id.

Returns

The node's position.

7.16.2.3 int is_node_alive (nodeid_t id)

Check wether a node is alive.

Parameters

id	the node id.

Returns

1 if the node is alive, 0 otherwise.

7.16.2.4 void node_kill (nodeid_t *id*)

Kill a node during the simulation.

Parameters

id	the node id.

7.17 options.h File Reference

User options declarations.

Defines

- #define CHANNELS_NUMBER 1

  Define the number of simulated radio channels.
- #define SNR_STEP 1

  Define the interference support policy.
- #define SNR_ERRORS 0

  Define the packet error policy.
- #define LOG_REPLAY
- #define LOG_APPLICATION
- #define LOG_MAC
- #define LOG_WORLDSENS

7.17.1 Detailed Description

User options declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file options.h.

7.17.2 Define Documentation

7.17.2.1 #define CHANNELS_NUMBER 1

Define the number of simulated radio channels.

Definition at line 14 of file options.h.

7.17.2.2 #define LOG_APPLICATION

Definition at line 32 of file options.h.

7.17.2.3 #define LOG_MAC

Definition at line 34 of file options.h.

7.17.2.4 #define LOG_REPLAY

Definition at line 31 of file options.h.

7.17.2.5 #define LOG_WORLDSENS

Definition at line 44 of file options.h.

7.17.2.6 #define SNR_ERRORS 0

Define the packet error policy.

0 = no error introduction in packet data, 1 = error introduction in the packet data according to the computed BER.

Definition at line 25 of file options.h.

7.17.2.7 #define SNR_STEP 1

Define the interference support policy.

0 = no interference, -1 = SINR computation for each packet byte, x = packet divided in x slices for SINR computation.

Definition at line 20 of file options.h.

7.18 packet.h File Reference

Packet declarations.

```
#include <include/types.h>
```

Functions

- void packet_dealloc (packet_t *packet)

  Deallocate a packet.
- packet_t * packet_clone (packet_t *packet)

  Duplicate a packet.
- packet_t * packet_create (call_t *c, int size, int real_size)

  Allocate a packet with real size argument.
- packet_t * packet_alloc (call_t *c, int size)

  Allocate a packet - DEPRECATED, use packet_create instead.

7.18.1 Detailed Description

Packet declarations.

Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

Date

2007

Definition in file packet.h.

7.18.2 Function Documentation

7.18.2.1 packet_t* packet_alloc (call_t * *c*, int *size*)

Allocate a packet - DEPRECATED, use packet_create instead.

Parameters

c	should be {-1, node id, -1}.
size	the data size of the packet (in bytes).

Returns

The newly allocated packet.

7.18.2.2 packet_t* packet_clone (packet_t * *packet*)

Duplicate a packet.

Data and other information are copied.

Parameters

packet	the packet to clone.

Returns

The cloned packet.

7.18.2.3 packet_t* packet_create (call_t * *c*, int *size*, int *real_size*)

Allocate a packet with real size argument.

Parameters

c	should be {-1, node id, -1}.
size	the data size of the packet (in bytes).
real_size	real size in bits of the packet.

Returns

The newly allocated packet.

7.18.2.4 void packet_dealloc (packet_t * *packet*)

Deallocate a packet.

Parameters

packet	the packet to dealloc.

7.19 param.h File Reference

```
#include <include/types.h>
```

Functions

- int get_param_x_position (char *value, double *position)
  Parse a x axis position parameter.
- int get_param_y_position (char *value, double *position)
  Parse a y axis position parameter.
- int get_param_z_position (char *value, double *position)
  Parse a z axis position parameter.

- int get_param_distance (char *value, double *distance)

  Parse a distance parameter.
- int get_param_double (char *value, double *res)

  Parse a double parameter.
- int get_param_double_range (char *value, double *res, double min, double max)

  Parse a double parameter.
- int get_param_time (char *value, uint64_t *time)

  Parse a time parameter.
- int get_param_integer (char *value, int *integer)

  Parse an integer parameter.
- int get_param_nodeid (char *value, nodeid_t *node, nodeid_t myself)

  Parse a node id parameter.
- int get_param_entity (char *value, entityid_t *entity)

  Parse an entity parameter.

7.19.1 Function Documentation

7.19.1.1 int get_param_distance (char * *value*, double * *distance*)

Parse a distance parameter.

Accept "random" parameter value.

Parameters

value	the parameter value.
distance	a pointer to a double that will be filled.

Returns

0 upon success, -1 otherwise.

7.19.1.2 int get_param_double (char * *value*, double * *res*)

Parse a double parameter.

Accept "random" parameter value.

Parameters

value	the parameter value.
res	a pointer to a double that will be filled.

Returns

0 upon success, -1 otherwise.

7.19.1.3 int get_param_double_range (char * *value,* double * *res,* double *min,* double *max*)

Parse a double parameter.

Accept "random" parameter value. The parsed value is checked to be in [min,max].

Parameters

value	the parameter value.
res	a pointer to a double that will be filled.
min	the minimum accepted value.
max	the maximum accepted value.

Returns

0 upon success, -1 otherwise.

7.19.1.4 int get_param_entity (char * *value,* entityid_t * *entity*)

Parse an entity parameter.

Parameters

value	the parameter value.
entity	a pointer to an entity id that will be filled.

Returns

0 upon success, -1 otherwise.

7.19.1.5 int get_param_integer (char * *value,* int * *integer*)

Parse an integer parameter.

Accept "random" parameter value.

Parameters

value	the parameter value.
integer	a pointer to an integer that will be filled.

Returns

0 upon success, -1 otherwise.

7.19.1.6 int get_param_nodeid (char * *value,* nodeid_t * *node,* nodeid_t *myself*)

Parse a node id parameter.

Accept "random" parameter value. The returned param value will be different than "myself".

Parameters

value	the parameter value.
node	a pointer to a node id that will be filled.
myself	a node id to exclude.

Returns

0 upon success, -1 otherwise.

7.19.1.7 int get_param_time (char * *value,* uint64_t * *time*)

Parse a time parameter.

Accept "random" parameter value.

Parameters

value	the parameter value.
time	a pointer to a uin64 that will be filled.

Returns

0 upon success, -1 otherwise.

7.19.1.8 int get_param_x_position (char * *value,* double * *position*)

Parse a x axis position parameter.

Accept "random" parameter value.

Parameters

value	the parameter value.
position	a pointer to a double that will be filled.

Returns

0 upon success, -1 otherwise.

7.19.1.9 int get_param_y_position (char * *value*, double * *position*)

Parse a y axis position parameter.

Accept "random" parameter value.

Parameters

value	the parameter value.
position	a pointer to a double that will be filled.

Returns

0 upon success, -1 otherwise.

7.19.1.10 int get_param_z_position (char * *value*, double * *position*)

Parse a z axis position parameter.

Accept "random" parameter value.

Parameters

value	the parameter value.
position	a pointer to a double that will be filled.

Returns

0 upon success, -1 otherwise.

7.20 probabilistic_distribution.h File Reference

Probabilistic distributions declarations.

Data Structures

- struct gaussian_args_s
- struct gaussian_tail_args_s
- struct bivariate_gaussian_args_s
- struct exponential_args_s
- struct laplace_args_s
- struct exppow_args_s

- struct cauchy_args_s
- struct rayleigh_s
- struct rayleigh_tail_s
- struct levy_alpha_stable_s
- struct gamma_args_s
- struct uniform_args_s
- struct lognormal_args_s
- struct chisq_args_s
- struct beta_args_s
- struct logistic_args_s
- struct pareto_args_s
- struct spherical_vector_2d_args_s
- struct spherical_vector_3d_args_s
- struct weibull_args_s
- struct gumbel_t1_args_s
- struct gumbel_t2_args_s
- struct poisson_args_s
- struct bernoulli_args_s
- struct binomial_args_s
- struct geometric_args_s
- struct hyper_geometric_args_s
- struct logarithmic_args_s

Defines

- #define GAUSSIAN 0
- #define GAUSSIAN_TAIL 1
- #define EXPONENTIAL 2
- #define LAPLACE 3
- #define EXPONENTIAL_POWER 4
- #define CAUCHY 5
- #define RAYLEIGH 6
- #define RAYLEIGH_TAIL 7
- #define LANDAU 8
- #define LEVY_ALPHA_STABLE 9
- #define GAMMA 10
- #define UNIFORM 11
- #define LOG_NORMAL 12
- #define CHI_SQUARED 13
- #define BETA 14
- #define LOGISTIC 15
- #define PARETO 16
- #define WEIBULL 17
- #define GUMBELL_1 18
- #define GUMBELL_2 19
- #define POISSON 20

- #define BERNOULLI 21
- #define BINOMIAL 22
- #define GEOMETRIC 23
- #define HYPERGEOMETRIC 24
- #define LOGARITHMIC 25

Typedefs

- typedef struct gaussian_args_s gaussian_args_t
- typedef struct gaussian_tail_args_s gaussian_tail_args_t
- typedef struct bivariate_gaussian_args_s bivariate_gaussian_args_t
- typedef struct exponential_args_s exponential_args_t
- typedef struct laplace_args_s laplace_args_t
- typedef struct exppow_args_s exppow_args_t
- typedef struct cauchy_args_s cauchy_args_t
- typedef struct rayleigh_s rayleigh_args_t
- typedef struct rayleigh_tail_s rayleigh_tail_args_t
- typedef struct levy_alpha_stable_s levy_alpha_stable_args_t
- typedef struct gamma_args_s gamma_args_t
- typedef struct uniform_args_s uniform_args_t
- typedef struct lognormal_args_s lognormal_args_t
- typedef struct chisq_args_s chisq_args_t
- typedef struct beta_args_s beta_args_t
- typedef struct logistic_args_s logistic_args_t
- typedef struct pareto_args_s pareto_args_t
- typedef struct spherical_vector_2d_args_s spherical_vector_2d_args_t
- typedef struct spherical_vector_3d_args_s spherical_vector_3d_args_t
- typedef struct weibull_args_s weibull_args_t
- typedef struct gumbel_t1_args_s gumbel_t1_args_t
- typedef struct gumbel_t2_args_s gumbel_t2_args_t
- typedef struct poisson_args_s poisson_args_t
- typedef struct bernoulli_args_s bernoulli_args_t
- typedef struct binomial_args_s binomial_args_t
- typedef struct geometric_args_s geometric_args_t
- typedef struct hyper_geometric_args_s hyper_geometric_args_t
- typedef struct logarithmic_args_s logarithmic_args_t
- typedef double(∗ distribution_function_t)(void ∗rng_id, void ∗parameters)

Functions

- distribution_function_t get_distribution_function_by_type (int type)

  Return a pointer to the distribution function according to type type of the distribution.

7.20.1 Detailed Description

Probabilistic distributions declarations.

Author

Quentin Lampin

Date

2009

Definition in file probabilistic_distribution.h.

7.20.2 Define Documentation

7.20.2.1 #define BERNOULLI 21

Definition at line 33 of file probabilistic_distribution.h.

7.20.2.2 #define BETA 14

Definition at line 26 of file probabilistic_distribution.h.

7.20.2.3 #define BINOMIAL 22

Definition at line 34 of file probabilistic_distribution.h.

7.20.2.4 #define CAUCHY 5

Definition at line 17 of file probabilistic_distribution.h.

7.20.2.5 #define CHI_SQUARED 13

Definition at line 25 of file probabilistic_distribution.h.

7.20.2.6 #define EXPONENTIAL 2

Definition at line 14 of file probabilistic_distribution.h.

7.20.2.7 #define EXPONENTIAL_POWER 4

Definition at line 16 of file probabilistic_distribution.h.

7.20.2.8 #define GAMMA 10

Definition at line 22 of file probabilistic_distribution.h.

7.20.2.9 #define GAUSSIAN 0

Definition at line 12 of file probabilistic_distribution.h.

7.20.2.10 #define GAUSSIAN_TAIL 1

Definition at line 13 of file probabilistic_distribution.h.

7.20.2.11 #define GEOMETRIC 23

Definition at line 35 of file probabilistic_distribution.h.

7.20.2.12 #define GUMBELL_1 18

Definition at line 30 of file probabilistic_distribution.h.

7.20.2.13 #define GUMBELL_2 19

Definition at line 31 of file probabilistic_distribution.h.

7.20.2.14 #define HYPERGEOMETRIC 24

Definition at line 36 of file probabilistic_distribution.h.

7.20.2.15 #define LANDAU 8

Definition at line 20 of file probabilistic_distribution.h.

7.20.2.16 #define LAPLACE 3

Definition at line 15 of file probabilistic_distribution.h.

7.20.2.17 #define LEVY_ALPHA_STABLE 9

Definition at line 21 of file probabilistic_distribution.h.

7.20.2.18 #define LOG_NORMAL 12

Definition at line 24 of file probabilistic_distribution.h.

7.20.2.19 #define LOGARITHMIC 25

Definition at line 37 of file probabilistic_distribution.h.

7.20.2.20 #define LOGISTIC 15

Definition at line 27 of file probabilistic_distribution.h.

7.20.2.21 #define PARETO 16

Definition at line 28 of file probabilistic_distribution.h.

7.20.2.22 #define POISSON 20

Definition at line 32 of file probabilistic_distribution.h.

7.20.2.23 #define RAYLEIGH 6

Definition at line 18 of file probabilistic_distribution.h.

7.20.2.24 #define RAYLEIGH_TAIL 7

Definition at line 19 of file probabilistic_distribution.h.

7.20.2.25 #define UNIFORM 11

Definition at line 23 of file probabilistic_distribution.h.

7.20.2.26 #define WEIBULL 17

Definition at line 29 of file probabilistic_distribution.h.

7.20.3 Typedef Documentation

7.20.3.1 typedef struct bernoulli_args_s bernoulli_args_t

7.20.3.2 typedef struct beta_args_s beta_args_t

7.20.3.3 typedef struct binomial_args_s binomial_args_t

7.20.3.4 typedef struct bivariate_gaussian_args_s bivariate_gaussian_args_t

7.20.3.5 typedef struct cauchy_args_s cauchy_args_t

7.20.3.6 typedef struct chisq_args_s chisq_args_t

7.20.3.7 typedef double(∗ distribution_function_t)(void ∗rng_id, void ∗parameters)

Definition at line 174 of file probabilistic_distribution.h.

7.20.3.8 typedef struct exponential_args_s exponential_args_t

7.20.3.9 typedef struct exppow_args_s exppow_args_t

7.20.3.10 typedef struct gamma_args_s gamma_args_t

7.20.3.11 typedef struct gaussian_args_s gaussian_args_t

7.20.3.12 typedef struct gaussian_tail_args_s gaussian_tail_args_t

7.20.3.13 typedef struct geometric_args_s geometric_args_t

7.20.3.14 typedef struct gumbel_t1_args_s gumbel_t1_args_t

7.20.3.15 typedef struct gumbel_t2_args_s gumbel_t2_args_t

7.20.3.16 typedef struct hyper_geometric_args_s hyper_geometric_args_t

7.20.3.17 typedef struct laplace_args_s laplace_args_t

7.20.3.18 typedef struct levy_alpha_stable_s levy_alpha_stable_args_t

7.20.3.19 typedef struct logarithmic_args_s logarithmic_args_t

7.20.3.20 typedef struct logistic_args_s logistic_args_t

7.20.3.21 typedef struct lognormal_args_s lognormal_args_t

7.20.3.22 typedef struct pareto_args_s pareto_args_t

7.20.3.23 typedef struct poisson_args_s poisson_args_t

7.20.3.24 typedef struct rayleigh_s rayleigh_args_t

7.20.3.25 typedef struct rayleigh_tail_s rayleigh_tail_args_t

7.20.3.26 typedef struct spherical_vector_2d_args_s spherical_vector_2d_args_t

7.20.3.27 typedef struct spherical_vector_3d_args_s spherical_vector_3d_args_t

7.20.3.28 typedef struct uniform_args_s uniform_args_t

7.20.3.29 typedef struct weibull_args_s weibull_args_t

7.20.4 Function Documentation

7.20.4.1 distribution_function_t get_distribution_function_by_type (int *type*)

Return a pointer to the distribution function according to type type of the distribution.

Returns

a function pointer

7.21 radio.h File Reference

Radio declarations.

```
#include <include/types.h>
```

Functions

- double radio_get_cs (call_t *c)
  Carrier Sense mechanism.
- double radio_get_noise (call_t *c)
  Clear Channel Assesment mechanism.
- double radio_get_power (call_t *c)
  Get the radio tx power.
- void radio_set_power (call_t *c, double power)
  Set the radio tx power.
- int radio_get_channel (call_t *c)
  Get the radio channel.
- void radio_set_channel (call_t *c, int channel)
  Set the radio channel.
- entityid_t radio_get_modulation (call_t *c)
  Get the radio modulation.
- void radio_set_modulation (call_t *c, entityid_t modulation)
  Set the radio modulation.
- uint64_t radio_get_Tb (call_t *c)
  Get the radio bandwidth.
- void radio_cs (call_t *c, packet_t *packet)

Notifies the radio with a new signal.

- void radio_set_sensibility (call_t *c, double sensibility)

  Set the radio sensibility.
- double radio_get_sensibility (call_t *c)

  Get the radio sensibility.
- void radio_sleep (call_t *c)

  Set the radio in sleep mode.
- void radio_wakeup (call_t *c)

  Set the radio in active mode.
- int radio_get_modulation_bit_per_symbol (call_t *c)

  get the number of bit per symbol for modulation associated
- uint64_t radio_get_Ts (call_t *c)

  Get the radio bandwidth.
- void radio_set_Ts (call_t *c, uint64_t Ts)

  Set the radio bandwidth.

7.21.1 Detailed Description

Radio declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file radio.h.

7.21.2 Function Documentation

7.21.2.1 void radio_cs (call_t * *c*, packet_t * *packet*)

Notifies the radio with a new signal.

Parameters

c	should be {radio id, node id, antenna id}.
packet	the new signal.

7.21.2.2 int radio_get_channel (call_t * *c*)

Get the radio channel.

Parameters

c	should be {radio id, node id, -1}.

Returns

The current radio channel.

7.21.2.3 double radio_get_cs (call_t * c)

Carrier Sense mechanism.

Parameters

c	should be {radio id, node id, -1}.

Returns

The signal strength of the currently received signal. MIN_DBM if no current signal.

7.21.2.4 entityid_t radio_get_modulation (call_t * c)

Get the radio modulation.

Parameters

c	should be {radio id, node id, -1}.

Returns

The current modulation entity id.

7.21.2.5 int radio_get_modulation_bit_per_symbol (call_t * c)

get the number of bit per symbol for modulation associated

Parameters

c	should be {radio id, node id, -1}.

Returns

the current number of bit per symbol

7.21.2.6 double radio_get_noise (call_t * c)

Clear Channel Assesment mechanism.

Parameters

c	should be {radio id, node id, -1}.

Returns

The noise strength on the radio interface. MIN_DBM if no noise.

7.21.2.7 double radio_get_power (call_t * c)

Get the radio tx power.

Parameters

c	should be {radio id, node id, -1}.

Returns

The current transmission power in dBm.

7.21.2.8 double radio_get_sensibility (call_t * c)

Get the radio sensibility.

Parameters

c	should be {radio id, node id, -1}.

Returns

The current radio sensibility in dBm.

7.21.2.9 uint64_t radio_get_Tb (call_t * c)

Get the radio bandwidth.

Parameters

c	should be {radio id, node id, -1}.

Returns

The current bandwidth, i.e. the time to transmit a bit in ns.

7.21.2.10 uint64_t radio_get_Ts (call_t * c)

Get the radio bandwidth.

Parameters

c	should be {radio id, node id, -1}.

Returns

The current bandwidth, i.e. the time to transmit a symbol in ns.

7.21.2.11 void radio_set_channel (call_t * c, int channel)

Set the radio channel.

Parameters

c	should be {radio id, node id, -1}.
channel	the new radio channel.

7.21.2.12 void radio_set_modulation (call_t * c, entityid_t modulation)

Set the radio modulation.

Parameters

c	should be {radio id, node id, -1}.
modulation	the new modulation entity id.

7.21.2.13 void radio_set_power (call_t * c, double power)

Set the radio tx power.

Parameters

c	should be {radio id, node id, -1}.
power	the new transmission power in dBm.

7.21.2.14 void radio_set_sensibility (call_t * c, double sensibility)

Set the radio sensibility.

Parameters

c	should be {radio id, node id, -1}.
sensibility	the new radio sensibility in dBm.

7.21.2.15 void radio_set_Ts (call_t * c, uint64_t Ts)

Set the radio bandwidth.

Ts must be a bit_per_symbol's multiple

Parameters

c	should be {radio id, node id, -1}.
Ts	the new bandwidth, i.e. the time to transmit a symbol in ns.

7.21.2.16 void radio_sleep (call_t * c)

Set the radio in sleep mode.

Parameters

c	should be {radio id, node id, -1}.

7.21.2.17 void radio_wakeup (call_t * c)

Set the radio in active mode.

Parameters

c	should be {radio id, node id, -1}.

7.22 rng.h File Reference

Random number generator declarations.

`#include <include/types.h> #include <gsl/gsl_rng.h>`

Defines

- #define DEFAULT_RNG 1

- #define MT19937 1
- #define RANLXS0 2
- #define RANLXS1 3
- #define RANLXS2 4
- #define RANLXD1 5
- #define RANLXD2 6
- #define RANLUX 7
- #define RANLUX389 8
- #define RNG_DEFAULT_RETRY_ATTEMPTS -1

Functions

- gsl_rng * get_rng_by_id (void *rng_id)

  retrieve RNG by ID.
- void * create_rng (int rng_type, unsigned long int seed)

  create RNG
- double get_random_distance_gsl (void *rng_id, int distribution_type, void *parameters)

  Return a random distance.
- double get_random_x_position_gsl (void *rng_id, int distribution_type, void *parameters)

  Return a random position on the x dimension.
- double get_random_y_position_gsl (void *rng_id, int distribution_type, void *parameters)

  Return a random position on the y dimension.
- double get_random_z_position_gsl (void *rng_id, int distribution_type, void *parameters)

  Return a random position on the z dimension.
- double get_random_double_gsl (void *rng_id, int distribution_type, void *parameters)

  Return a random double value in [0,1[.
- double get_random_double_range_gsl (void *rng_id, int distribution_type, void *parameters, double min, double max)

  Return a random double value in [min,max[.
- int get_random_integer_gsl (void *rng_id, int distribution_type, void *parameters)

  Return a random integer value.
- int get_random_integer_range_gsl (void *rng_id, int distribution_type, void *parameters, int min, int max)

  Return a random integer value in [min,max].
- uint64_t get_random_time_gsl (void *rng_id, int distribution_type, void *parameters)

  Return a random time.
- uint64_t get_random_time_range_gsl (void *rng_id, int distribution_type, void *parameters, uint64_t min, uint64_t max)

Return a random time in [min,max].

- nodeid_t get_random_node_gsl (void *rng_id, int distribution_type, void *parameters, nodeid_t exclusion)

  Return a random node id different from a specified one.

- double get_random_distance (void)

  ALL THE FOLLOWING FUNCTION ARE USED BY THE WSNET KERNEL WITH THE DEFAULT RNG.

- double get_random_x_position (void)

  Return a random position on the x dimension.

- double get_random_y_position (void)

  Return a random position on the y dimension.

- double get_random_z_position (void)

  Return a random position on the z dimension.

- double get_random_double (void)

  Return a random double value in [0,1[.

- double get_random_double_range (double min, double max)

  Return a random double value in [min,max[.

- int get_random_integer (void)

  Return a random integer value.

- int get_random_integer_range (int min, int max)

  Return a random integer value in [min,max].

- uint64_t get_random_time (void)

  Return a random time in [0,simulation_end].

- uint64_t get_random_time_range (uint64_t min, uint64_t max)

  Return a random time in [min,max].

- nodeid_t get_random_node (nodeid_t exclusion)

  Return a random node id different from a specified one.

7.22.1 Detailed Description

Random number generator declarations.

Author

Quentin Lampin

Date

2009

Definition in file rng.h.

7.22.2 Define Documentation

7.22.2.1 #define DEFAULT_RNG 1

Definition at line 15 of file rng.h.

7.22.2.2 #define MT19937 1

Definition at line 18 of file rng.h.

7.22.2.3 #define RANLUX 7

Definition at line 24 of file rng.h.

7.22.2.4 #define RANLUX389 8

Definition at line 25 of file rng.h.

7.22.2.5 #define RANLXD1 5

Definition at line 22 of file rng.h.

7.22.2.6 #define RANLXD2 6

Definition at line 23 of file rng.h.

7.22.2.7 #define RANLXS0 2

Definition at line 19 of file rng.h.

7.22.2.8 #define RANLXS1 3

Definition at line 20 of file rng.h.

7.22.2.9 #define RANLXS2 4

Definition at line 21 of file rng.h.

7.22.2.10 #define RNG_DEFAULT_RETRY_ATTEMPTS -1

Definition at line 28 of file rng.h.

7.22.3 Function Documentation

7.22.3.1 void* create_rng (int *rng_type,* unsigned long int *seed*)

create RNG

Parameters

rng_type	the type of the RNG to instanciate
seed	the seed of the RNG

Returns

id of the newly created RNG

7.22.3.2 double get_random_distance (void)

ALL THE FOLLOWING FUNCTION ARE USED BY THE WSNET KERNEL WITH THE DEFAULT RNG.

Return a random distance.

Returns

A random distance.

7.22.3.3 double get_random_distance_gsl (void * *rng_id,* int *distribution_type,* void * *parameters*)

Return a random distance.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters

Returns

A random distance.

7.22.3.4 double get_random_double (void)

Return a random double value in [0,1[.

Returns

A random double value in [0,1[.

7.22.3.5 double get_random_double_gsl (void * *rng_id*, int *distribution_type*, void * *parameters*)

Return a random double value in [0,1[.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters

Returns

A random double value in [0,1[.

7.22.3.6 double get_random_double_range (double *min*, double *max*)

Return a random double value in [min,max[.

Parameters

min	the min value that can be drawn.
max	the max value that can be drawn.

Returns

A random double value in [min,max[.

7.22.3.7 double get_random_double_range_gsl (void * *rng_id*, int *distribution_type*, void * *parameters*, double *min*, double *max*)

Return a random double value in [min,max[.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters
min	the min value that can be drawn.
max	the max value that can be drawn.

Returns

A random double value in [min,max[.

7.22.3.8 int get_random_integer (void)

Return a random integer value.

Returns

A random integer value.

7.22.3.9 int get_random_integer_gsl (void * *rng_id,* int *distribution_type,* void * *parameters*)

Return a random integer value.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters

Returns

A random integer value.

7.22.3.10 int get_random_integer_range (int *min,* int *max*)

Return a random integer value in [min,max].

Parameters

min	the min value that can be drawn.
max	the max value that can be drawn.

Returns

A random integer value.

7.22.3.11 int get_random_integer_range_gsl (void * *rng_id,* int *distribution_type,* void * *parameters,* int *min,* int *max*)

Return a random integer value in [min,max].

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters
min	the min value that can be drawn.
max	the max value that can be drawn.

Returns

A random integer value.

7.22.3.12 nodeid_t get_random_node (nodeid_t *exclusion*)

Return a random node id different from a specified one.

Parameters

exclusion	the node identifier we do not want to be returned.

Returns

A random node identifier.

7.22.3.13 nodeid_t get_random_node_gsl (void * *rng_id*, int *distribution_type*, void * *parameters*, nodeid_t *exclusion*)

Return a random node id different from a specified one.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters
exclusion	the node identifier we do not want to be returned.

Returns

A random node identifier.

7.22.3.14 uint64_t get_random_time (void)

Return a random time in [0,simulation_end].

Returns

A random time [0,simulation_end].

7.22.3.15 uint64_t get_random_time_gsl (void * *rng_id,* int *distribution_type,* void * *parameters*)

Return a random time.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters

Returns

A random time in ns.

7.22.3.16 uint64_t get_random_time_range (uint64_t *min,* uint64_t *max*)

Return a random time in [min,max].

Parameters

min	the min value that can be drawn.
max	the max value that can be drawn.

Returns

A random time in [min,max].

7.22.3.17 uint64_t get_random_time_range_gsl (void * *rng_id,* int *distribution_type,* void * *parameters,* uint64_t *min,* uint64_t *max*)

Return a random time in [min,max].

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters
min	the min value that can be drawn.
max	the max value that can be drawn.

Returns

A random time in [min,max].

7.22.3.18 double get_random_x_position (void)

Return a random position on the x dimension.

Returns

A random position on the x dimension.

7.22.3.19 double get_random_x_position_gsl (void * *rng_id*, int *distribution_type*, void * *parameters*)

Return a random position on the x dimension.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters

Returns

A random position on the x dimension.

7.22.3.20 double get_random_y_position (void)

Return a random position on the y dimension.

Returns

A random position on the y dimension.

7.22.3.21 double get_random_y_position_gsl (void * *rng_id*, int *distribution_type*, void * *parameters*)

Return a random position on the y dimension.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters

Returns

A random position on the y dimension.

7.22.3.22 double get_random_z_position (void)

Return a random position on the z dimension.

Returns

A random position on the z dimension.

7.22.3.23 double get_random_z_position_gsl (void * *rng_id*, int *distribution_type*, void * *parameters*)

Return a random position on the z dimension.

Parameters

rng_id	ID of the RNG to use.
distribution_type	type of the probability distribution
parameters	the probability distribution parameters

Returns

A random position on the z dimension.

7.22.3.24 gsl_rng* get_rng_by_id (void * *rng_id*)

retrieve RNG by ID.

Parameters

rng_id	ID of the rng to retrieve.

Returns

pointer to the RNG.

7.23 scheduler.h File Reference

Scheduler declarations.

```
#include <include/types.h>
```

Data Structures

- struct _event

  A scheduler event.

Typedefs

- typedef struct _event event_t

  A scheduler event.

Functions

- event_t * scheduler_add_callback (uint64_t clock, call_t *c, callback_t callback, void *arg)

  Schedule the callback of a function at a given time.
- void scheduler_delete_callback (call_t *c, event_t *event)

  Delete an event from the Scheduler.

7.23.1 Detailed Description

Scheduler declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file scheduler.h.

7.23.2 Typedef Documentation

7.23.2.1 event_t

A scheduler event.

7.23.3 Function Documentation

7.23.3.1 event_t* scheduler_add_callback (uint64_t *clock,* call_t * *c,* callback_t *callback,* void * *arg*)

Schedule the callback of a function at a given time.

Parameters

clock	time of the callback.
c	the call parameter given to the callback function.
callback	the function that is called back.
arg	a paramater that given to the callback function.

Returns

An opaque object that references the callback-associated event.

7.23.3.2 void scheduler_delete_callback (call_t * *c*, event_t * *event*)

Delete an event from the Scheduler.

Parameters

c	the call parameter given to the callback function.
event	a paramater that describe the event we want to delete.

7.24 sodas.h File Reference

SOrted DAta Structure module declarations.

Typedefs

- typedef int(* sodas_compare_t)(void *key0, void *key1)

  The prototype of a comparison function.

Functions

- int sodas_init (void)

  Initialize the sodas module.
- void * sodas_create (sodas_compare_t compare)

  Create an empty sodas structure.
- void sodas_destroy (void *sodas)

  Destroy a sodas structure.
- void sodas_insert (void *sodas, void *key, void *data)

  Insert an object in the sodas structure.
- void * sodas_pop (void *sodas)

  Remove the first object from the sodas structure.
- void * sodas_delete (void *sodas, void *key)

  Remove a particular object from the sodas structure.

- void * sodas_see_first (void *sodas)

  Show the first object in the sodas structure.

7.24.1 Detailed Description

SOrted DAta Structure module declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file sodas.h.

7.24.2 Typedef Documentation

7.24.2.1 sodas_compare_t

The prototype of a comparison function.

Definition at line 14 of file sodas.h.

7.24.3 Function Documentation

7.24.3.1 void* sodas_create (sodas_compare_t *compare*)

Create an empty sodas structure.

Parameters

comapre	the comparison function used to sort the objects.

Returns

An opaque pointer to the sodas structure.

7.24.3.2 void* sodas_delete (void * *sodas*, void * *key*)

Remove a particular object from the sodas structure.

Parameters

sodas	the opaque pointer to the data structure.
key	the key of the object to remove.

7.24.3.3 void sodas_destroy (void * *sodas*)

Destroy a sodas structure.

Objects in the structure are not deallocated.

Parameters

sodas	the opaque pointer to the sodas structure.

7.24.3.4 int sodas_init (void)

Initialize the sodas module.

Done by the wsnet core.

Returns

0 if success, -1 otherwise.

7.24.3.5 void sodas_insert (void * *sodas,* void * *key,* void * *data*)

Insert an object in the sodas structure.

Parameters

sodas	the opaque pointer to the sodas structure.
key	the object key.
data	the object to insert.

7.24.3.6 void* sodas_pop (void * *sodas*)

Remove the first object from the sodas structure.

Parameters

sodas	the opaque pointer to the data structure.

Returns

The first object.

7.24.3.7 void* sodas_see_first (void * *sodas*)

Show the first object in the sodas structure.

The object is not removed.

Parameters

sodas	the opaque pointer to the data structure.

Returns

The first object.

7.25 spadas.h File Reference

Space PArtitioning DAta Structure module declarations.

```
#include <include/types.h>
```

Functions

- int spadas_init (void)

  Initialize the spadas module.
- void * spadas_create (position_t *area, double range)

  Create an empty spadas structure.
- void spadas_destroy (void *spadas)

  Destroy a spadas structure.
- void spadas_insert (void *spadas, void *data, position_t *position)

  Insert an object in the spadas structure.
- void spadas_update (void *spadas, void *data, position_t *n_position, position_t *o_position)

  Update an object position in the spadas structure.
- void spadas_delete (void *spadas, void *data, position_t *position)

  Remove an object from the spadas structure.
- void * spadas_rangesearch (void *spadas, void *data, position_t *position, double range)

  Return all objects close to a given position.

7.25.1 Detailed Description

Space PArtitioning DAta Structure module declarations.

Author

Guillaume Chelius & Elyes Ben Hamida

Date

2007

Definition in file spadas.h.

7.25.2 Function Documentation

7.25.2.1 void* spadas_create (position_t * area, double range)

Create an empty spadas structure.

Parameters

area	the area where objects of the spadas will be.
range	a distance value used for spadas optimization.

Returns

An opaque pointer to the spadas structure.

7.25.2.2 void spadas_delete (void * spadas, void * data, position_t * position)

Remove an object from the spadas structure.

Parameters

spadas	the opaque pointer to the spadas structure.
data	the object to remove.
position	the object position.

7.25.2.3 void spadas_destroy (void * spadas)

Destroy a spadas structure.

Objects in the structure are not deallocated.

Parameters

spadas	the opaque pointer to the spadas structure.

7.25.2.4 int spadas_init (void)

Initialize the spadas module.

Done by the wsnet core.

Returns

0 if success, -1 otherwise.

7.25.2.5 void spadas_insert (void * *spadas*, void * *data*, position_t * *position*)

Insert an object in the spadas structure.

Parameters

spadas	the opaque pointer to the spadas structure.
data	the object to insert.
position	the object position.

7.25.2.6 void* spadas_rangesearch (void * *spadas*, void * *data*, position_t * *position*, double *range*)

Return all objects close to a given position.

Parameters

spadas	the opaque pointer to the spadas structure.
data	an object which can be excluded from the search.
position	the central position.
range	distance to the central position.

Returns

A das containing all objects that are at distance <= range from the central position.

7.25.2.7 void spadas_update (void * *spadas*, void * *data*, position_t * *n_position*, position_t * *o_position*)

Update an object position in the spadas structure.

Parameters

spadas	the opaque pointer to the spadas structure.
data	the object.
n_position	the new object position.
o_position	the old object position.

7.26 timer.h File Reference

Generic timer.

Data Structures

- struct qtimer_s

timer structure parameters: parameters for next_trigger functions (eg: period for periodic timer) conditional_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback_function: function to callback when the timer triggers.

- struct exponential_s
- struct uniform_random_s

Typedefs

- typedef struct qtimer_s qtimer_t

  timer structure parameters: parameters for next_trigger functions (eg: period for periodic timer) conditional_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback_function: function to callback when the timer triggers.
- typedef struct exponential_s exponential_parameters_t
- typedef struct uniform_random_s uniform_random_parameters_t

Functions

- uint64_t periodic_trigger (call_t *c, void *timer_id)

  qtimer and not timer because timer_t is already defined in /usr/lib/time.c so I added my first name initial ;)
- uint64_t exponential_trigger (call_t *c, void *timer_id)
- uint64_t uniform_random_trigger (call_t *c, void *timer_id)
- int never_stop (call_t *c, void *timer_id)
- void * create_timer (call_t *c, void *callback_function, void *conditional_end, void *next_trigger, void *trigger_parameters)
- void * start_timer (void *timer_id, uint64_t delay)
- void destroy_timer (void *timer_id)
- void change_parameter (void *timer_id, void *new_parameters)
- int timer_init (void)
- void timer_clean (void)
- qtimer_t * fetch_timer (void *timer_id)

7.26.1 Detailed Description

Generic timer.

Author

Quentin Lampin

Date

2009

Definition in file timer.h.

7.26.2 Typedef Documentation

7.26.2.1 typedef struct exponential_s exponential_parameters_t

7.26.2.2 typedef struct qtimer_s qtimer_t

timer structure parameters: parameters for next_trigger functions (eg: period for periodic timer) conditional_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback_function: function to callback when the timer triggers.

7.26.2.3 typedef struct uniform_random_s uniform_random_parameters_t

7.26.3 Function Documentation

7.26.3.1 void change_parameter (void * *timer_id,* void * *new_parameters*)

7.26.3.2 void* create_timer (call_t * *c,* void * *callback_function,* void * *conditional_end,* void * *next_trigger,* void * *trigger_parameters*)

7.26.3.3 void destroy_timer (void * *timer_id*)

7.26.3.4 uint64_t exponential_trigger (call_t * *c,* void * *timer_id*)

7.26.3.5 qtimer_t* fetch_timer (void * *timer_id*)

7.26.3.6 int never_stop (call_t * *c,* void * *timer_id*)

7.26.3.7 uint64_t periodic_trigger (call_t * *c,* void * *timer_id*)

qtimer and not timer because timer_t is already defined in /usr/lib/time.c so I added my first name initial ;)

7.26.3.8 void* start_timer (void * *timer_id,* uint64_t *delay*)

7.26.3.9 void timer_clean (void)

7.26.3.10 int timer_init (void)

7.26.3.11 uint64_t uniform_random_trigger (call_t * *c,* void * *timer_id*)

7.27 types.h File Reference

Type declarations.

`#include <stdlib.h> #include <inttypes.h> #include <math.h>`

Data Structures

- struct _array

  An array of integers containing its size.
- struct _position

  A position in the 3D space.
- struct _angle

  An angle in the 3D space.
- struct _destination

  A packet destination.
- struct _call

  A parameter that identifies who we are calling and who has called us.
- struct _packet

  A radio packet.
- struct _ioctl_message
- struct _param

  A parameter for the "init" and "setnode" entity functions.

Typedefs

- typedef struct _array array_t

  An array of integers containing its size.
- typedef int nodeid_t

  A node identifier.
- typedef int entityid_t

  An entity identifier.
- typedef int measureid_t

  A measure identifier.
- typedef int packetid_t

  A packet identifier.
- typedef struct _position position_t

  A position in the 3D space.
- typedef struct _angle angle_t

  An angle in the 3D space.
- typedef struct _destination destination_t

  A packet destination.
- typedef struct _call call_t

  A parameter that identifies who we are calling and who has called us.
- typedef struct _packet packet_t

  A radio packet.
- typedef struct _ioctl_message ioctl_message_t

  An ioctl message.
- typedef struct _param param_t

  A parameter for the "init" and "setnode" entity functions.
- typedef int(∗ callback_t)(call_t ∗c, void ∗arg)

  Prototype of a callback function.

7.27.1 Detailed Description

Type declarations.

Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

Date

2007

Definition in file types.h.

7.27.2 Typedef Documentation

7.27.2.1 angle_t

An angle in the 3D space.

7.27.2.2 array_t

An array of integers containing its size.

7.27.2.3 call_t

A parameter that identifies who we are calling and who has called us.

Kind of a self pointer.

7.27.2.4 callback_t

Prototype of a callback function.

Definition at line 187 of file types.h.

7.27.2.5 destination_t

A packet destination.

May be a node address or a geographical position.

7.27.2.6 entityid_t

An entity identifier.

Definition at line 42 of file types.h.

7.27.2.7 ioctl_message_t

An ioctl message.

7.27.2.8 measureid_t

A measure identifier.

Definition at line 51 of file types.h.

7.27.2.9 nodeid_t

A node identifier.

Definition at line 33 of file types.h.

7.27.2.10 packet_t

A radio packet.

7.27.2.11 packetid_t

A packet identifier.

Definition at line 60 of file types.h.

7.27.2.12 param_t

A parameter for the "init" and "setnode" entity functions.

7.27.2.13 position_t

A position in the 3D space.

7.28 worldsens_debug.h File Reference

Defines

- #define WSNET_S_DBG 0
- #define WSNET_S_EXC_DBG 0
- #define WSNET_S_DBG_OUT(x...) fprintf(stderr, x)
- #define WSNET_S_DBG_DBG(x...) do { } while (0)
- #define WSNET_S_DBG_EXC(x...) do { } while (0)

7.28.1 Define Documentation

7.28.1.1 #define WSNET_S_DBG 0

Definition at line 20 of file worldsens_debug.h.

7.28.1.2 #define WSNET_S_DBG_DBG(*x...*) do { } while (0)

Definition at line 33 of file worldsens_debug.h.

7.28.1.3 #define WSNET_S_DBG_EXC(*x...*) do { } while (0)

Definition at line 39 of file worldsens_debug.h.

7.28.1.4 #define WSNET_S_DBG_OUT(*x...*) fprintf(stderr, x)

Definition at line 28 of file worldsens_debug.h.

7.28.1.5 #define WSNET_S_EXC_DBG 0

Definition at line 21 of file worldsens_debug.h.

7.29 worldsens_pkt.h File Reference

Worldsens packet format.

```
#include <inttypes.h>
```

Data Structures

- struct _worldsens_c_header
- struct _worldsens_c_connect_req
- struct _worldsens_c_sync_ack
- struct _worldsens_c_byte_tx
- struct _worldsens_c_measure_req
- struct _worldsens_c_disconnect
- struct _worldsens_s_header
- struct _worldsens_s_connect_rsp_ok
- struct _worldsens_s_connect_rsp_nok
- struct _worldsens_s_sync_release
- struct _worldsens_s_backtrack
- struct _worldsens_s_sync_reminder
- struct _worldsens_s_byte_rx

- struct _worldsens_s_byte_sr_rx
- struct _worldsens_s_measure_rsp
- struct _worldsens_s_measure_sr_rsp
- struct _worldsens_s_killsim
- struct _worldsens_s_kill
- union _worldsens_pkt

Defines

- #define __PACKED__ __attribute__((packed))
- #define WORLDSENS_MAX_PKTLENGTH (sizeof(union _worldsens_pkt))
- #define WORLDSENS_MAX_MODELS_SIZE 1200

Typedefs

- typedef uint8_t ws_pkt_type
- typedef uint32_t ws_id_node
- typedef int64_t ws_id_resource
- typedef uint64_t ws_id_rp
- typedef uint64_t ws_id_seq
- typedef uint64_t ws_frequency
- typedef uint64_t ws_power
- typedef uint64_t ws_measure
- typedef uint64_t ws_sinr
- typedef uint64_t ws_time
- typedef uint8_t ws_data

Enumerations

- enum woldsens_pkt_type { WORLDSENS_UNKNOWN = 0, WORLDSENS_C_CONNECT_REQ, WORLDSENS_C_SYNC_ACK, WORLDSENS_C_BYTE_TX, WORLDSENS_C_MEASURE_REQ, WORLDSENS_C_DISCONNECT, WORLDSENS_S_CONNECT_RSP_OK, WORLDSENS_S_CONNECT_RSP_NOK, WORLDSENS_S_SYNC_RELEASE, WORLDSENS_S_SYNC_REMINDER, WORLDSENS_S_BACKTRACK, WORLDSENS_S_BYTE_RX, WORLDSENS_S_BYTE_SR_RX, WORLDSENS_S_MEASURE_RSP, WORLDSENS_S_MEASURE_SR_RSP, WORLDSENS_S_KILLSIM, WORLDSENS_S_KILL, WORLDSENS_LASTID }

Functions

- int worldsens_packet_hton (union _worldsens_pkt *pkt)
- int worldsens_packet_ntoh (union _worldsens_pkt *pkt)
- int worldsens_packet_dump (union _worldsens_pkt *pkt)

7.29.1 Detailed Description

Worldsens packet format.

Author

Guillaume Chelius, Antoine Fraboulet, Loic Lemaitre

Date

2007

Definition in file worldsens_pkt.h.

7.29.2 Define Documentation

7.29.2.1 #define __PACKED__ __attribute__((packed))

Definition at line 14 of file worldsens_pkt.h.

7.29.2.2 #define WORLDSENS_MAX_MODELS_SIZE 1200

Definition at line 124 of file worldsens_pkt.h.

7.29.2.3 #define WORLDSENS_MAX_PKTLENGTH (sizeof(union _worldsens_pkt))

Definition at line 16 of file worldsens_pkt.h.

7.29.3 Typedef Documentation

7.29.3.1 typedef uint8_t ws_data

Definition at line 69 of file worldsens_pkt.h.

7.29.3.2 typedef uint64_t ws_frequency

Definition at line 63 of file worldsens_pkt.h.

7.29.3.3 typedef uint32_t ws_id_node

Definition at line 56 of file worldsens_pkt.h.

7.29.3.4 typedef int64_t ws_id_resource

Definition at line 57 of file worldsens_pkt.h.

7.29.3.5 typedef uint64_t ws_id_rp

Definition at line 58 of file worldsens_pkt.h.

7.29.3.6 typedef uint64_t ws_id_seq

Definition at line 59 of file worldsens_pkt.h.

7.29.3.7 typedef uint64_t ws_measure

Definition at line 65 of file worldsens_pkt.h.

7.29.3.8 typedef uint8_t ws_pkt_type

Definition at line 54 of file worldsens_pkt.h.

7.29.3.9 typedef uint64_t ws_power

Definition at line 64 of file worldsens_pkt.h.

7.29.3.10 typedef uint64_t ws_sinr

Definition at line 66 of file worldsens_pkt.h.

7.29.3.11 typedef uint64_t ws_time

Definition at line 68 of file worldsens_pkt.h.

7.29.4 Enumeration Type Documentation

7.29.4.1 enum woldsens_pkt_type

Enumerator:

WORLDSENS_UNKNOWN

WORLDSENS_C_CONNECT_REQ

WORLDSENS_C_SYNC_ACK

WORLDSENS_C_BYTE_TX

WORLDSENS_C_MEASURE_REQ

WORLDSENS_C_DISCONNECT

WORLDSENS_S_CONNECT_RSP_OK

WORLDSENS_S_CONNECT_RSP_NOK

WORLDSENS_S_SYNC_RELEASE
WORLDSENS_S_SYNC_REMINDER
WORLDSENS_S_BACKTRACK
WORLDSENS_S_BYTE_RX
WORLDSENS_S_BYTE_SR_RX
WORLDSENS_S_MEASURE_RSP
WORLDSENS_S_MEASURE_SR_RSP
WORLDSENS_S_KILLSIM
WORLDSENS_S_KILL
WORLDSENS_LASTID

Definition at line 23 of file worldsens_pkt.h.

7.29.5 Function Documentation

7.29.5.1 int worldsens_packet_dump (union _worldsens_pkt * *pkt*)

7.29.5.2 int worldsens_packet_hton (union _worldsens_pkt * *pkt*)

7.29.5.3 int worldsens_packet_ntoh (union _worldsens_pkt * *pkt*)