SpiNNMan Documentation

Release

April 23, 2015

Contents

Used to communicate with a SpiNNaker Board. The main part of this package is the `spinnman.transceiver.Transceiver` class. This can be used to send and receive packets in various SpiNNaker formats, depending on what connections are available.

CHAPTER 1

Functional Requirements

- Connect to and communicate with a machine using a number of different connections.
- Boot a machine with the expected version of the software.
  - If the machine is already booted but the version is not the version expected, an exception will be thrown.
- Check the version of the software which the machine is booted with.
- Query the state of the machine to determine:
  - What the current state of the machine is in terms of the chips and cores available, the SDRAM available on the chips and the which links are available between which chips.
  - What external links to the host exist (and separately add the discovered links to the set of links used to communicate with the machine).
  - What is running on the machine and where, and what the current status of those processes are.
  - How many cores are in a given state.
  - What is in the IOBUF buffers.
  - What the current routing entries for a given router are.
  - What the routing status counter values are.
- Load application binaries on to the machine, either to individual cores or via a "flood-fill" mechanism to multiple cores simultaneously (which may be a subset of the cores on a subset of the chips).
- Write data to SDRAM, either on an individual chip, or via a "flood-fill" mechanism to multiple chips simultaneously.
- Send a signal to an application.
- Read data from SDRAM on an individual chip.
- Send and receive SpiNNaker packets where the connections allow this.
  - If no connection supports this packet type, an exception is thrown.
  - The user should be able to select which connection is used. Selection of a connection which does not support the traffic type will also result in an exception.
- Send and receive SCP and SDP packets where the connections allow this.
  - If no connection supports the packet type, an exception is thrown.
  - The user should be able to select which connection is used. Selection of a connection which does not support the traffic type will also result in an exception.

- It should be possible to call any of the functions simultaneously, including the same function more than once.
  - Where possible, multiple connections should be used to overlap calls.
  - The functions should not return until they have confirmed that any messages sent have been received, and any responses have been received.
  - Functions should not respond with the result of a different function.
  - Functions can further sub-divide the call into a number of separate calls that can be divided across the available connections, so long as the other requirements are met.
- More than one machine can be connected to the same host.
  - Once the subset of connections has been worked out for each machine, the operation of these machines should be independent.

CHAPTER 2

Use Cases

- `boot_board()` and `get_scamp_version()` are used to ensure that the board is booted correctly before starting a simulation.
- `get_machine_details()` is used to get a representation of the current state of the machine, which is used to decide where executables are to be run on the board for a particular simulation, where any external peripherals are connected, and how messages between the executables and/or the external peripherals are to be routed
- `write_memory()` and `execute()` are used to write parameters and execute executables on the board
- `send_signal()` is used to send a signal which starts, stops or pauses a simulation.
- `get_core_status_count()` is used to determine if a simulation is complete or has gone into an error state.
- `get_iobuf()`, `get_cpu_information()` and `get_router_diagnostics()` are used to diagnose a problem with a simulation
- `read_memory()` is used to read some statistics recorded in SDRAM after a simulation
- `set_ip_tag()` and `SCPListener` are used to set up an IP Tag on the machine and receive transmitted SCP packets from the machine
- `send_multicast()` and `send_scp_message()` are used to inject packets in to the machine from the host

CHAPTER 3

Contents

3.1 spinnman package

3.1.1 Subpackages

spinnman.connections package

Subpackages

spinnman.connections.abstract_classes package

Subpackages

spinnman.connections.abstract_classes.udp_receivers package

Submodules

spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_command_receiver module

class spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_command_receiver

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A receiver of SCP messages

Abstract Methods

is_udp_eieio_command_receiver()

Methods

receive_eieio_command_message([timeout])	Receives an eieio message from this connection.

Detailed Methods

is_udp_eieio_command_receiver()

receive_eieio_command_message(*timeout=None*)
Receives an eieio message from this connection. Blocks until a message has been received, or a timeout occurs.

Parameters **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message
- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid SCP message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the SCP message is invalid

spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_data_receiver module

class spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_data_receiver.**Ab**
Bases: spinnman.connections.abstract_classes.abstract_eieio_receiver.AbstractEIEIOReceive

A receiver of SCP messages

Abstract Methods

is_udp_eieio_receiver()

Methods

is_eieio_receiver()	
receive_eieio_message([timeout])	Receives an eieio message from this connection.

Detailed Methods

is_eieio_receiver()

is_udp_eieio_receiver()

receive_eieio_message(*timeout=None*)
Receives an eieio message from this connection. Blocks until a message has been received, or a timeout occurs.

Parameters **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message
- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid SCP message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the SCP message is invalid

spinnman.connections.abstract_classes.udp_receivers.abstract_udp_scp_receiver module

class spinnman.connections.abstract_classes.udp_receivers.abstract_udp_scp_receiver.**AbstractU**

Bases: spinnman.connections.abstract_classes.abstract_scp_receiver.AbstractSCPReceiver

A receiver of SCP messages

Abstract Methods

is_udp_scp_receiver()

Methods

receive_scp_response(scp_response[, timeout])	Receives an SCP message from this connection.

Detailed Methods

is_udp_scp_receiver()

receive_scp_response(*scp_response*, *timeout=None*)

Receives an SCP message from this connection. Blocks until a message has been received, or a timeout occurs.

Parameters

- **scp_response** – The response to fill in
- **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Rtype scp_response spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message
- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received

- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid SCP message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the SCP message is invalid

spinnman.connections.abstract_classes.udp_receivers.abstract_udp_sdp_receiver module

class spinnman.connections.abstract_classes.udp_receivers.abstract_udp_sdp_receiver.**AbstractU**

Bases: spinnman.connections.abstract_classes.abstract_sdp_receiver.AbstractSDPReceiver

A receiver of SDP messages

Abstract Methods

is_udp_sdp_reciever()

Methods

receive_sdp_message([timeout])	Receives an SDP message from this connection.

Detailed Methods

is_udp_sdp_reciever()

receive_sdp_message(*timeout=None*)

Receives an SDP message from this connection. Blocks until the message has been received, or a timeout occurs.

Parameters **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Returns The received SDP message

Return type spinnman.messages.sdp.sdp_message.SDPMessage

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message
- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid SDP message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the SDP message is invalid

spinnman.connections.abstract_classes.udp_senders package

Submodules

spinnman.connections.abstract_classes.udp_senders.abstract_udp_eieio_command_sender module

class spinnman.connections.abstract_classes.udp_senders.abstract_udp_eieio_command_sender.**Abs**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A receiver of SCP messages

Abstract Methods

is_udp_eieio_command_sender()

Methods

send_eieio_command_message(eieio_command_message)	Sends an SDP message down this connection

Detailed Methods

is_udp_eieio_command_sender()

send_eieio_command_message(*eieio_command_message*)

Sends an SDP message down this connection

Parameters **eieio_command_message** (*spinnman.messages.eieio.eieio_command_message.EIEIOCommandMessage* – The eieio message to be sent

Returns Nothing is returned

Return type None

Raises **spinnman.exceptions.SpinnmanIOException** If there is an error sending the message

spinnman.connections.abstract_classes.udp_senders.abstract_udp_eieio_sender module

class spinnman.connections.abstract_classes.udp_senders.abstract_udp_eieio_sender.**AbstractUDP**

Bases: spinnman.connections.abstract_classes.abstract_eieio_sender.AbstractEIEIOSender

A receiver of SCP messages

Abstract Methods

is_udp_eieio_sender()

Methods

send_eieio_message(eieio_message)	Sends an SDP message down this connection

Detailed Methods

is_udp_eieio_sender()

send_eieio_message(*eieio_message*)

Sends an SDP message down this connection

Parameters **eieio_message** (spinnman.messages.eieio.abstract_messages.abstract_eieio_mes – The eieio message to be sent

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error sending the message
- **spinnman.exceptions.SpinnmanInvalidParameterTypeException** – If the message passed to be sent does not inherit from spinnman.messages.eieio.abstract_messages.abstract_eieio_message.AbstractEI

spinnman.connections.abstract_classes.udp_senders.abstract_udp_scp_sender module

class spinnman.connections.abstract_classes.udp_senders.abstract_udp_scp_sender.**AbstractUDPSC**

Bases: spinnman.connections.abstract_classes.abstract_scp_sender.AbstractSCPSender

A sender of SCP messages

Abstract Methods

is_udp_scp_sender()

Methods

send_scp_request(scp_request)	Sends an SCP request down this connection

Detailed Methods

is_udp_scp_sender()

send_scp_request(*scp_request*)

Sends an SCP request down this connection

Messages must have the following properties:

•source_port is None or 7

•source_cpu is None or 31

•source_chip_x is None or 0

•source_chip_y is None or 0

tag in the message is optional - if not set the default set in the constructor will be used. sequence in the message is optional - if not set (sequence number last assigned + 1) % 65536 will be used

Parameters **scp_request** (spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest – message packet to send

Returns Nothing is returned

Return type None

Raises spinnman.exceptions.SpinnmanIOException If there is an error sending the message

spinnman.connections.abstract_classes.udp_senders.abstract_udp_sdp_sender module

class spinnman.connections.abstract_classes.udp_senders.abstract_udp_sdp_sender.**AbstractUDPSD**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A sender of SDP messages

Abstract Methods

is_udp_sdp_sender()

Methods

send_sdp_message(sdp_message)	Sends an SDP message down this connection

Detailed Methods

is_udp_sdp_sender()

send_sdp_message(*sdp_message*)

Sends an SDP message down this connection

Parameters **sdp_message** (*spinnman.messages.sdp.sdp_message.SDPMessage*) – The SDP message to be sent

Returns Nothing is returned

Return type None

Raises spinnman.exceptions.SpinnmanIOException If there is an error sending the message

Submodules

spinnman.connections.abstract_classes.abstract_callbackable_connection module

class spinnman.connections.abstract_classes.abstract_callbackable_connection.**AbstractCallback**

Bases: object

Abstract Methods

deregister_callback(callback)
register_callback(callback, traffic_type)

Detailed Methods

deregister_callback(*callback*)

register_callback(*callback*, *traffic_type*)

spinnman.connections.abstract_classes.abstract_connection module

class spinnman.connections.abstract_classes.abstract_connection.**AbstractConnection**

Bases: object

An abstract connection to the SpiNNaker board over some medium

Abstract Methods

close()	Closes the connection
is_connected()	Determines if the medium is connected at this point in time

Detailed Methods

close()

Closes the connection

Returns Nothing is returned

Return type None

Raises None No known exceptions are raised

is_connected()

Determines if the medium is connected at this point in time

Returns True if the medium is connected, False otherwise

Return type bool

Raises spinnman.exceptions.SpinnmanIOException If there is an error when determining the connectivity of the medium

spinnman.connections.abstract_classes.abstract_eieio_receiver module

class spinnman.connections.abstract_classes.abstract_eieio_receiver.**AbstractEIEIOReceiver**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

Abstract Methods

is_eieio_receiver()	
receive_eieio_message([timeout])	

Detailed Methods

is_eieio_receiver()

receive_eieio_message(*timeout=None*)

spinnman.connections.abstract_classes.abstract_eieio_sender module

class spinnman.connections.abstract_classes.abstract_eieio_sender.**AbstractEIEIOSender**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A receiver of SCP messages

Abstract Methods

is_eieio_sender()	
send_eieio_message(eieio_message)	Sends an SDP message down this connection

Detailed Methods

is_eieio_sender()

send_eieio_message(*eieio_message*)
Sends an SDP message down this connection

Parameters **eieio_message** (spinnman.messages.eieio.eieio_data_message.EIEIODataMessage) – The eieio message to be sent

Returns Nothing is returned

Return type None

Raises spinnman.exceptions.SpinnmanIOException If there is an error sending the message

spinnman.connections.abstract_classes.abstract_multicast_receiver module

class spinnman.connections.abstract_classes.abstract_multicast_receiver.**AbstractMulticastRece**
Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A receiver of Multicast messages

Abstract Methods

get_input_chips()	Get a list of chips which identify the chips from which this receiver can receive rec
receive_multicast_message([timeout])	Receives a multicast message from this connection.

Detailed Methods

get_input_chips()
Get a list of chips which identify the chips from which this receiver can receive receive packets directly

Returns An iterable of tuples of (x, y) where x is the x-coordinate of the chip and y is the y-coordinate of the chip

Return type iterable of (int, int)

Raises None No known exceptions are raised

receive_multicast_message(*timeout=None*)
Receives a multicast message from this connection. Blocks until a message has been received, or a timeout occurs.

Parameters **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Returns a multicast message

Return type spinnman.messages.multicast_message.MulticastMessage

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message
- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid multicast message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the multicast message is invalid

spinnman.connections.abstract_classes.abstract_multicast_sender module

class spinnman.connections.abstract_classes.abstract_multicast_sender.**AbstractMulticastSender**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A sender of Multicast messages

Abstract Methods

get_input_chips()	Get a list of chips which identify the chips to which this sender can send mult
send_multicast_message(multicast_message)	Sends a SpiNNaker multicast message using this connection

Detailed Methods

get_input_chips()

Get a list of chips which identify the chips to which this sender can send multicast packets directly

Returns An iterable of tuples of (x, y) where x is the x-coordinate of the chip and y is the y-coordinate of the chip

Return type iterable of (int, int)

Raises None No known exceptions are raised

send_multicast_message(*multicast_message*)

Sends a SpiNNaker multicast message using this connection

Parameters **multicast_message** (spinnman.messages.multicast_message.MulticastMessage) – The message to be sent

Returns Nothing is returned

Return type None

Raises **spinnman.exceptions.SpinnmanIOException** If there is an error sending the message

spinnman.connections.abstract_classes.abstract_scp_receiver module

class spinnman.connections.abstract_classes.abstract_scp_receiver.**AbstractSCPReceiver**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A receiver of SCP messages

Abstract Methods

Continued on next page

Table 3.23 – continued from previous page

is_scp_receiver()	
receive_scp_response(scp_response[, timeout])	Receives an SCP message from this connection.

Detailed Methods

is_scp_receiver()

receive_scp_response(*scp_response*, *timeout=None*)

Receives an SCP message from this connection. Blocks until a message has been received, or a timeout occurs.

Parameters

- **scp_response** – The response to fill in
- **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Rtype scp_response spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message
- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid SCP message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the SCP message is invalid

spinnman.connections.abstract_classes.abstract_scp_sender module

class spinnman.connections.abstract_classes.abstract_scp_sender.**AbstractSCPSender**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A sender of SCP messages

Abstract Methods

is_udp_scp_sender()	
send_scp_request(scp_request)	Sends an SCP request down this connection

Detailed Methods

is_udp_scp_sender()

send_scp_request(*scp_request*)

Sends an SCP request down this connection

Messages must have the following properties:

- source_port is None or 7
- source_cpu is None or 31
- source_chip_x is None or 0
- source_chip_y is None or 0

tag in the message is optional - if not set the default set in the constructor will be used. sequence in the message is optional - if not set (sequence number last assigned + 1) % 65536 will be used

Parameters **scp_request** (spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest – message packet to send

Returns Nothing is returned

Return type None

Raises spinnman.exceptions.SpinnmanIOException If there is an error sending the message

spinnman.connections.abstract_classes.abstract_sdp_receiver module

class spinnman.connections.abstract_classes.abstract_sdp_receiver.**AbstractSDPReceiver**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A receiver of SDP messages

Abstract Methods

is_sdp_reciever()	
receive_sdp_message([timeout])	Receives an SDP message from this connection.

Detailed Methods

is_sdp_reciever()

receive_sdp_message(*timeout=None*)

Receives an SDP message from this connection. Blocks until the message has been received, or a timeout occurs.

Parameters **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Returns The received SDP message

Return type spinnman.messages.sdp.sdp_message.SDPMessage

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message
- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid SDP message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the SDP message is invalid

spinnman.connections.abstract_classes.abstract_sdp_sender module

class spinnman.connections.abstract_classes.abstract_sdp_sender.**AbstractSDPSender**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A sender of SDP messages

Abstract Methods

is_sdp_sender()	
send_sdp_message(sdp_message)	Sends an SDP message down this connection

Detailed Methods

is_sdp_sender()

send_sdp_message(*sdp_message*)

Sends an SDP message down this connection

Parameters **sdp_message** (*spinnman.messages.sdp.sdp_message.SDPMessage*) – The SDP message to be sent

Returns Nothing is returned

Return type None

Raises **spinnman.exceptions.SpinnmanIOException** If there is an error sending the message

spinnman.connections.abstract_classes.abstract_spinnaker_boot_receiver module

class spinnman.connections.abstract_classes.abstract_spinnaker_boot_receiver.**AbstractSpinnake**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A receiver of Spinnaker boot messages

Abstract Methods

receive_boot_message([timeout])	Receives a boot message from this connection.

Detailed Methods

receive_boot_message(*timeout=None*)

Receives a boot message from this connection. Blocks until a message has been received, or a timeout occurs.

Parameters **timeout** (*int*) – The time in seconds to wait for the message to arrive; if not specified, will wait forever, or until the connection is closed

Returns a boot message

Return type spinnman.messages.spinnaker_boot.spinnaker_boot_message.SpinnakerBootM

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error receiving the message

- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the received packet is not a valid spinnaker boot message
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the spinnaker boot message is invalid

spinnman.connections.abstract_classes.abstract_spinnaker_boot_sender module

class spinnman.connections.abstract_classes.abstract_spinnaker_boot_sender.**AbstractSpinnakerB**

Bases: spinnman.connections.abstract_classes.abstract_connection.AbstractConnection

A sender of Spinnaker Boot messages

Abstract Methods

send_boot_message(boot_message)	Sends a SpiNNaker boot message using this connection

Detailed Methods

send_boot_message(*boot_message*)

Sends a SpiNNaker boot message using this connection

Parameters **boot_message** (spinnman.messages.spinnaker_boot.spinnaker_boot_message.Spin – The message to be sent

Returns Nothing is returned

Return type None

Raises **spinnman.exceptions.SpinnmanIOException** If there is an error sending the message

spinnman.connections.abstract_classes.abstract_udp_connection module

class spinnman.connections.abstract_classes.abstract_udp_connection.**AbstractUDPConnection**(*loca*
lo-
*cal_*
re-
mot
re-
mot

Bases: object

Parameters

- **local_host** (*str or None*) – The local host name or ip address to bind to. If not specified defaults to bind to all interfaces, unless remote_host is specified, in which case binding is _done to the ip address that will be used to send packets
- **local_port** (*int*) – The local port to bind to, between 1025 and 65535. If not specified, defaults to a random unused local port
- **remote_host** (*str or None*) – The remote host name or ip address to send packets to. If not specified, the socket will be available for listening only, and will throw and exception if used for sending

- **remote_port** – The remote port to send packets to. If remote_host is None, this is ignored.

Raises spinnman.exceptions.SpinnmanIOException If there is an error setting up the communication channel

Attributes

can_send	a helper property method to check if this connection can send
local_ip_address	The local IP address to which the connection is bound.
local_port	The local port to which the connection is bound.
remote_ip_address	The remote ip address to which the connection is connected.
remote_port	The remote port to which the connection is connected.

Abstract Methods

connection_type()	method to help identify the connection
supports_sends_message(message)	helper method to verify if the connection can deal with this message

Methods

close()	See spinnman.connections.abstract_connection.AbstractConnection.close()
is_connected()	See spinnman.connections.AbstractConnection.abstract_connection.is_connected

Detailed Methods

close()
See spinnman.connections.abstract_connection.AbstractConnection.close()

connection_type()
method to help identify the connection :return:

is_connected()
See spinnman.connections.AbstractConnection.abstract_connection.is_connected()

supports_sends_message(*message*)
helper method to verify if the connection can deal with this message format

Parameters **message** –

Returns

spinnman.connections.listeners package

Subpackages

spinnman.connections.listeners.queuers package

Submodules

spinnman.connections.listeners.queuers.abstract_port_queuer module

class spinnman.connections.listeners.queuers.abstract_port_queuer.**AbstractPortQueuer**

Bases: threading.Thread

A Queue for packets received

Attributes

daemon	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.

Methods

get_packet()	Get the next packet from the queue
run()	
stop()	Stop the thread

Detailed Methods

get_packet()

Get the next packet from the queue

Returns The next packet, or None if the queue has been stopped

run()

stop()

Stop the thread

spinnman.connections.listeners.queuers.callback_worker module

class spinnman.connections.listeners.queuers.callback_worker.**CallbackWorker**

Bases: object

Methods

call_callback(callback, packet)

Detailed Methods

static call_callback(*callback*, *packet*)

spinnman.connections.listeners.queuers.eieio_command_port_queuer module

class spinnman.connections.listeners.queuers.eieio_command_port_queuer.**EIEIOCommandPortQueuer**

Bases: spinnman.connections.listeners.queuers.abstract_port_queuer.AbstractPortQueuer

Queuer of EIEIO Commands

Attributes

daemon	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.

spinnman.connections.listeners.queuers.eieio_data_port_queuer module

class spinnman.connections.listeners.queuers.eieio_data_port_queuer.**EIEIODataPortQueuer**(*connec*

Bases: spinnman.connections.listeners.queuers.abstract_port_queuer.AbstractPortQueuer

Queuer of EIEIO Data

Attributes

daemon	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.

spinnman.connections.listeners.queuers.scp_port_queuer module

class spinnman.connections.listeners.queuers.scp_port_queuer.**SCPPortQueuer**(*connection*)

Bases: spinnman.connections.listeners.queuers.abstract_port_queuer.AbstractPortQueuer

Queuer of SCP Data

Attributes

daemon	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.

spinnman.connections.listeners.queuers.sdp_port_queuer module

class spinnman.connections.listeners.queuers.sdp_port_queuer.**SDPPortQueuer**(*connection*)

Bases: spinnman.connections.listeners.queuers.abstract_port_queuer.AbstractPortQueuer

Queuer of SDP Messages

Attributes

daemon	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.

spinnman.connections.listeners.queuers.udp_port_queuer module

class spinnman.connections.listeners.queuers.udp_port_queuer.**UDPPortQueuer**(*connection*)

Bases: spinnman.connections.listeners.queuers.abstract_port_queuer.AbstractPortQueuer

Queuer of Raw UDP messages

Attributes

daemon	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.

Submodules

spinnman.connections.listeners.port_listener module

class spinnman.connections.listeners.port_listener.**PortListener**(*callback*, *queuer*, *no_threads=5*)

Bases: threading.Thread

Attributes

daemon	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.

Methods

	deregister_callback(callback)
	register_callback(callback)
	run()
	set_port(port)
	stop()

Detailed Methods

deregister_callback(*callback*)

register_callback(*callback*)

run()

set_port(*port*)

stop()

spinnman.connections.listeners.scp_listener module

class spinnman.connections.listeners.scp_listener.**SCPListener**(*scp_receiver*, *response_class*, *callback*, *error_callback=None*)

Bases: threading.Thread

Listens for SCP packets received from a connection, calling a callback function with received packets

Parameters

- **scp_receiver** (spinnman.connections.abstract_scp_receiver.AbstractSCPReceiver) – The SCP Receiver to receive packets from

- **response_class** (class of implementation of `spinnman.messages.scp.abstract_scp_response.Abst` – The SCP response
- **callback** (function(`spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse` – The callback function to call on reception of each packet; the function should take one parameter, which is the SCP packet received
- **error_callback** (*function(Exception, str)*) – The callback function to call if there is an error receiving a packet; the function should take two parameters: * The exception received * A message indicating what the problem was

Raises **spinnman.exceptions.SpinnmanInvalidParameterException** If the callback or the error_callback do not take the expected number of arguments

Attributes

`daemon`	A boolean value indicating whether this thread is a daemon thread (True) or not (False).
`ident`	Thread identifier of this thread or None if it has not been started.
`name`	A string used for identification purposes only.

Methods

`run`()	Overridden method of Thread that runs this listener
`start`()	Starts listening and sending callbacks
`stop`()	Stops the reception of packets

Detailed Methods

`run`()
Overridden method of Thread that runs this listener

`start`()
Starts listening and sending callbacks

Returns Nothing is returned

Return type None

Raises None No known exceptions are raised

`stop`()
Stops the reception of packets

Returns Nothing is returned

Return type None

Raises None No known exceptions are raised

spinnman.connections.udp_packet_connections package

Submodules

spinnman.connections.udp_packet_connections.eieio_command_connection module

class spinnman.connections.udp_packet_connections.eieio_command_connection.**EieioCommandConnec**

Bases: spinnman.connections.abstract_classes.abstract_udp_connection.AbstractUDPConnectio
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_command_receive
spinnman.connections.abstract_classes.udp_senders.abstract_udp_eieio_command_sender.Ab
spinnman.connections.abstract_classes.abstract_callbackable_connection.AbstractCallbac

Attributes

can_send	a helper property method to check if this connection can send
local_ip_address	The local IP address to which the connection is bound.
local_port	The local port to which the connection is bound.
remote_ip_address	The remote ip address to which the connection is connected.
remote_port	The remote port to which the connection is connected.

Methods

close()	
connection_type()	
deregister_callback(callback)	
is_udp_eieio_command_receiver()	
is_udp_eieio_command_sender()	
register_callback(callback[, traffic_type])	
supports_sends_message(message)	

Detailed Methods

close()

connection_type()

deregister_callback(*callback*)

is_udp_eieio_command_receiver()

is_udp_eieio_command_sender()

register_callback(*callback*, *traffic_type=<TRAFFIC_TYPE.EIEIO_COMMAND: 4>*)

supports_sends_message(*message*)

spinnman.connections.udp_packet_connections.iptag_connection module

class spinnman.connections.udp_packet_connections.iptag_connection.**IPTagConnection**(*local_host=No*
lo-
cal_port=Non
re-
mote_host=No
re-
mote_port=54

Bases: spinnman.connections.abstract_classes.abstract_udp_connection.AbstractUDPConnectio
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_scp_receiver.Abstract

spinnman.connections.abstract_classes.udp_receivers.abstract_udp_sdp_receiver.Abstract
spinnman.connections.abstract_classes.abstract_callbackable_connection.AbstractCallbac

Parameters

- **local_host** (*str*) – The local host name or ip address to bind to. If not specified defaults to bind to all interfaces, unless remote_host is specified, in which case binding is _done to the ip address that will be used to send packets
- **local_port** (*int*) – The local port to bind to, between 1025 and 65535. If not specified, defaults to a random unused local port
- **remote_host** (*str*) – The remote host name or ip address to send packets to. If not specified, the socket will be available for listening only, and will throw and exception if used for sending
- **remote_port** – The remote port to send packets to. If remote_host is None, this is ignored.

Raises spinnman.exceptions.SpinnmanIOException If there is an error setting up the communication channel

Attributes

can_send	a helper property method to check if this connection can send
local_ip_address	The local IP address to which the connection is bound.
local_port	The local port to which the connection is bound.
remote_ip_address	The remote ip address to which the connection is connected.
remote_port	The remote port to which the connection is connected.

Methods

close()	
connection_type()	
deregister_callback(callback)	
is_scp_receiver()	
is_sdp_reciever()	
is_udp_scp_receiver()	
is_udp_sdp_reciever()	
register_callback(callback, traffic_type)	
supports_sends_message(message)	

Detailed Methods

close()

connection_type()

deregister_callback(*callback*)

is_scp_receiver()

is_sdp_reciever()

is_udp_scp_receiver()

is_udp_sdp_reciever()

register_callback(*callback*, *traffic_type*)

supports_sends_message(*message*)

spinnman.connections.udp_packet_connections.reverse_iptag_connection module

class spinnman.connections.udp_packet_connections.reverse_iptag_connection.**ReverseIPTagConnec**

Bases: spinnman.connections.abstract_classes.abstract_udp_connection.AbstractUDPConnectio
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_data_receiver.A
spinnman.connections.abstract_classes.udp_senders.abstract_udp_eieio_sender.AbstractUD
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_command_receive
spinnman.connections.abstract_classes.udp_senders.abstract_udp_eieio_command_sender.Ab

Attributes

can_send	a helper property method to check if this connection can send
local_ip_address	The local IP address to which the connection is bound.
local_port	The local port to which the connection is bound.
remote_ip_address	The remote ip address to which the connection is connected.
remote_port	The remote port to which the connection is connected.

Methods

connection_type()	
is_eieio_receiver()	
is_eieio_sender()	
is_udp_eieio_command_receiver()	
is_udp_eieio_command_sender()	
is_udp_eieio_receiver()	
is_udp_eieio_sender()	
receive_raw()	
send_raw(message)	sends a raw udp packet
supports_sends_message(message)	

Detailed Methods

connection_type()

is_eieio_receiver()

is_eieio_sender()

is_udp_eieio_command_receiver()

is_udp_eieio_command_sender()

is_udp_eieio_receiver()

is_udp_eieio_sender()

receive_raw()

send_raw(*message*)
sends a raw udp packet :param message: the message sent in the udp packet :return: None

supports_sends_message(*message*)

spinnman.connections.udp_packet_connections.stripped_iptag_connection module

class spinnman.connections.udp_packet_connections.stripped_iptag_connection.**StrippedIPTagConn**

Bases: spinnman.connections.abstract_classes.abstract_udp_connection.AbstractUDPConnectio
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_data_receiver.A
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_eieio_command_receive
spinnman.connections.abstract_classes.abstract_callbackable_connection.AbstractCallbac

Parameters

- **local_host** (*str*) – The local host name or ip address to bind to. If not specified defaults to bind to all interfaces, unless remote_host is specified, in which case binding is _done to the ip address that will be used to send packets
- **local_port** (*int*) – The local port to bind to, between 1025 and 65535. If not specified, defaults to a random unused local port
- **remote_host** (*str*) – The remote host name or ip address to send packets to. If not specified, the socket will be available for listening only, and will throw and exception if used for sending
- **remote_port** – The remote port to send packets to. If remote_host is None, this is ignored.

Raises spinnman.exceptions.SpinnmanIOException If there is an error setting up the communication channel

Attributes

can_send	a helper property method to check if this connection can send
local_ip_address	The local IP address to which the connection is bound.
local_port	The local port to which the connection is bound.
remote_ip_address	The remote ip address to which the connection is connected.
remote_port	The remote port to which the connection is connected.

Methods

close()	
connection_type()	
deregister_callback(callback)	
is_udp_eieio_command_receiver()	

Continued on next page

Table 3.51 – continued from previous page

is_udp_eieio_receiver()	
recieve_raw(timeout)	
register_callback(callback, traffic_type)	
supports_sends_message(message)	

Detailed Methods

close()

connection_type()

deregister_callback(*callback*)

is_udp_eieio_command_receiver()

is_udp_eieio_receiver()

recieve_raw(*timeout*)

register_callback(*callback*, *traffic_type*)

supports_sends_message(*message*)

spinnman.connections.udp_packet_connections.udp_boot_connection module

class spinnman.connections.udp_packet_connections.udp_boot_connection.**UDPBootConnection**(*local_h* *lo-* *cal_po* *re-* *mote_h* *re-* *mote_p*

Bases: spinnman.connections.abstract_classes.abstract_udp_connection.AbstractUDPConnectio spinnman.connections.abstract_classes.abstract_spinnaker_boot_sender.AbstractSpinnaker spinnman.connections.abstract_classes.abstract_spinnaker_boot_receiver.AbstractSpinnak

A connection to the spinnaker board that uses UDP to for booting

Parameters

- **local_host** (*str*) – The local host name or ip address to bind to. If not specified defaults to bind to all interfaces, unless remote_host is specified, in which case binding is _done to the ip address that will be used to send packets
- **local_port** (*int*) – The local port to bind to, between 1025 and 65535. If not specified, defaults to a random unused local port
- **remote_host** (*str*) – The remote host name or ip address to send packets to. If not specified, the socket will be available for listening only, and will throw and exception if used for sending
- **remote_port** – The remote port to send packets to. If remote_host is None, this is ignored.

Raises spinnman.exceptions.SpinnmanIOException If there is an error setting up the communication channel

Attributes

can_send	a helper property method to check if this connection can send
local_ip_address	The local IP address to which the connection is bound.
local_port	The local port to which the connection is bound.
remote_ip_address	The remote ip address to which the connection is connected.
remote_port	The remote port to which the connection is connected.

Methods

connection_type()	
receive_boot_message([timeout])	See spinnman.connections.abstract_spinnaker_boot_receiver.Ab
send_boot_message(boot_message)	See spinnman.connections.abstract_spinnaker_boot_sender.Abst
supports_sends_message(message)	

Detailed Methods

connection_type()

receive_boot_message(*timeout=None*)
See spinnman.connections.abstract_spinnaker_boot_receiver.AbstractSpinnakerBootRecei

send_boot_message(*boot_message*)
See spinnman.connections.abstract_spinnaker_boot_sender.AbstractSpinnakerBootSender.

supports_sends_message(*message*)

spinnman.connections.udp_packet_connections.udp_spinnaker_connection module

class spinnman.connections.udp_packet_connections.udp_spinnaker_connection.**UDPSpinnakerConnec**

Bases: spinnman.connections.abstract_classes.abstract_udp_connection.AbstractUDPConnectio
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_sdp_receiver.Abstract
spinnman.connections.abstract_classes.udp_senders.abstract_udp_sdp_sender.AbstractUDPS
spinnman.connections.abstract_classes.udp_senders.abstract_udp_scp_sender.AbstractUDPS
spinnman.connections.abstract_classes.udp_receivers.abstract_udp_scp_receiver.Abstract

Attributes

can_send	a helper property method to check if this connection can send
chip_x	
chip_y	
local_ip_address	The local IP address to which the connection is bound.
local_port	The local port to which the connection is bound.
remote_ip_address	The remote ip address to which the connection is connected.
remote_port	The remote port to which the connection is connected.

Methods

connection_type()	
is_scp_receiver()	
is_sdp_reciever()	
is_udp_scp_receiver()	
is_udp_scp_sender()	
is_udp_sdp_reciever()	
is_udp_sdp_sender()	
supports_sends_message(message)	

Detailed Methods

connection_type()

is_scp_receiver()

is_sdp_reciever()

is_udp_scp_receiver()

is_udp_scp_sender()

is_udp_sdp_reciever()

is_udp_sdp_sender()

supports_sends_message(*message*)

spinnman.data package

Submodules

spinnman.data.abstract_byte_reader module

class spinnman.data.abstract_byte_reader.**AbstractByteReader**

Bases: object

An abstract reader of bytes. Note that due to endianness concerns, the methods of this reader should be used directly for the appropriate data type being read; e.g. an int should be written using read_int rather than calling read_byte 4 times unless a specific endianness is being achieved.

Abstract Methods

is_at_end()	returns true if the reader is currently at the end of the byte
read_byte()	Reads the next byte
read_int()	Read the next four bytes as in int value
read_long()	Reads the next eight bytes as a long value
read_short()	Reads the next two bytes as a short value

Methods

read_bytes([size])	Reads an array of bytes

Detailed Methods

is_at_end()
returns true if the reader is currently at the end of the byte reader

Returns returns true if the reader is currently at the end of the byte

reader false otherwise :rtype: bool

read_byte()
Reads the next byte

Returns A byte

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are no more bytes to read

read_bytes(*size=None*)
Reads an array of bytes

Parameters **size** (*int*) – The number of bytes to read, or None to read all of the remaining bytes

Returns An array of bytes

Return type bytearray

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read the requested size. Note that if there are no more bytes and size is None, an empty array will be returned

read_int()
Read the next four bytes as in int value

Returns An int

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read an int

read_long()
Reads the next eight bytes as a long value

Returns A long

Return type long

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read a long

read_short()
Reads the next two bytes as a short value

Returns A short

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read a short

spinnman.data.abstract_byte_writer module

class spinnman.data.abstract_byte_writer.**AbstractByteWriter**
Bases: object

An abstract writer of bytes. Note that due to endianness concerns, the methods of this writer should be used directly for the appropriate data type being written; e.g. an int should be written using write_int rather than calling write_byte 4 times with the parts of the int unless a specific endianness is being achieved.

Abstract Methods

get_n_bytes_written()	Determines how many bytes have been written in total
write_byte(byte_value)	Writes the lowest order byte of the given value
write_int(int_value)	Writes a four byte value
write_long(long_value)	Writes an eight byte value
write_short(short_value)	Writes the two lowest order bytes of the given value

Methods

write_bytes(byte_iterable)	Writes a set of bytes

Detailed Methods

get_n_bytes_written()
Determines how many bytes have been written in total

Returns The number of bytes written

Return type int

Raises None No known exception is raised

write_byte(*byte_value*)
Writes the lowest order byte of the given value

Parameters **byte_value** (*int*) – The byte to write

Returns Nothing is returned

Return type None

Raises IOError If there is an error writing to the stream

write_bytes(*byte_iterable*)
Writes a set of bytes

Parameters **byte_iterable** (*iterable of bytes*) – The bytes to write

Returns Nothing is returned

Return type None

Raises IOError If there is an error writing to the stream

write_int(*int_value*)
Writes a four byte value

Parameters **int_value** (*int*) – The integer to write

Returns Nothing is returned

Return type None

Raises IOError If there is an error writing to the stream

write_long(*long_value*)
Writes an eight byte value

Parameters **long_value** (*long*) – The long to write

Returns Nothing is returned

Return type None

Raises IOError If there is an error writing to the stream

write_short(*short_value*)
Writes the two lowest order bytes of the given value

Parameters **short_value** (*int*) – The short to write

Returns Nothing is returned

Return type None

Raises IOError If there is an error writing to the stream

spinnman.data.abstract_data_reader module

class spinnman.data.abstract_data_reader.**AbstractDataReader**
Bases: object

Abstract Methods

read(n_bytes)	Read a number of bytes from the underlying stream
readall()	Read the rest of the bytes from the underlying stream
readinto(array)	Read a number of bytes into an array from the underlying stream

Detailed Methods

read(*n_bytes*)
Read a number of bytes from the underlying stream

Parameters **n_bytes** (*int*) – The number of bytes to read.

Returns The bytes read from the underlying stream. May be less than requested.

Return type bytearray

Raises IOError If there is an error obtaining the bytes

readall()
Read the rest of the bytes from the underlying stream

Returns The bytes read

Return type bytearray

Raises IOError If there is an error obtaining the bytes

readinto(*array*)
Read a number of bytes into an array from the underlying stream

Parameters **array** (*bytearray*) – An array into which the bytes are to be read

Returns The number of bytes written in to the array

Return type int

Raises IOError If there is an error obtaining the bytes

spinnman.data.big_endian_byte_array_byte_reader module

class spinnman.data.big_endian_byte_array_byte_reader.**BigEndianByteArrayByteReader**(*data*)
Bases: spinnman.data.abstract_byte_reader.AbstractByteReader

A byte reader that reads from a byte array using big endian notation

Parameters **data** (*bytearray*) – The byte array to read the data from

Methods

is_at_end()	
read_byte()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_byte()
read_bytes([size])	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_bytes()
read_int()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_int()
read_long()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_long()
read_short()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_short()

Detailed Methods

is_at_end()

read_byte()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_byte()

read_bytes(*size=None*)
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_bytes()

read_int()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_int()

read_long()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_long()

read_short()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_short()

spinnman.data.big_endian_byte_array_byte_writer module

class spinnman.data.big_endian_byte_array_byte_writer.**BigEndianByteArrayByteWriter**

Bases: spinnman.data.abstract_byte_writer.AbstractByteWriter

A byte writer that writes to a byte array using big endian notation

Attributes

data	The data that was written

Methods

get_n_bytes_written()	See spinnman.data.abstract_byte_writer.AbstractByteWriter.get_n_byte
write_byte(byte_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_byte
write_bytes(byte_iterable)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_byte
write_int(int_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_int(
write_long(long_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_long
write_short(short_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_shor

Detailed Methods

get_n_bytes_written()
See spinnman.data.abstract_byte_writer.AbstractByteWriter.get_n_bytes_written()

write_byte(*byte_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_byte()

write_bytes(*byte_iterable*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_bytes()

write_int(*int_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_int()

write_long(*long_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_long()

write_short(*short_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_short()

spinnman.data.file_data_reader module

Classes

FileDataReader(filename)	A reader that can read data from a file
FileIO	file(name: str[, mode: str]) -> file IO object

class spinnman.data.file_data_reader.**FileDataReader**(*filename*)

Bases: spinnman.data.abstract_data_reader.AbstractDataReader

A reader that can read data from a file

Parameters **filename** (*str*) – The file to read

Raises **spinnman.exceptions.SpinnmanIOException** If the file cannot found or opened for reading

Methods

close()	Closes the file
read(n_bytes)	See spinnman.data.abstract_data_reader.AbstractDataReader.read()
readall()	See spinnman.data.abstract_data_reader.AbstractDataReader.readall()
readinto(data)	See spinnman.data.abstract_data_reader.AbstractDataReader.readinto()

Detailed Methods

close()
Closes the file

Returns Nothing is returned:

Return type None

Raises **spinnman.exceptions.SpinnmanIOException** If the file cannot be closed

read(*n_bytes*)
See spinnman.data.abstract_data_reader.AbstractDataReader.read()

readall()
See spinnman.data.abstract_data_reader.AbstractDataReader.readall()

readinto(*data*)
See spinnman.data.abstract_data_reader.AbstractDataReader.readinto()

class spinnman.data.file_data_reader.**FileIO**
Bases: _io._RawIOBase

file(name: str[, mode: str]) -> file IO object

Open a file. The mode can be 'r', 'w' or 'a' for reading (default), writing or appending. The file will be created if it doesn't exist when opened for writing or appending; it will be truncated when opened for writing. Add a '+' to the mode to allow simultaneous reading and writing.

Attributes

closed	True if the file is closed
closefd	True if the file descriptor will be closed
mode	String giving the file mode
next	x.next() -> the next value, or raise StopIteration
closed	True if the file is closed
next	x.next() -> the next value, or raise StopIteration
closed	True if the file is closed
next	x.next() -> the next value, or raise StopIteration

Methods

close(() -> None. Close the file.)	A closed file cannot be used for further I/O operations.
fileno(() -> int. “file descriptor”.)	This is needed for lower-level file interfaces, such the fcntl module.
flush	Flush write buffers, if applicable.
isatty(...)	
read(...)	Only makes one system call, so less data may be returned than requested In non-
readable(...)	
readall(...)	In non-blocking mode, returns as much as is immediately available, or None if n
readinto(() -> Same as RawIOBase.readinto().)	
readline	Read and return a line from the stream.
readlines	Return a list of lines from the stream.
seek((offset: int[, ...)	Argument offset is a byte count.
seekable(...)	
tell(() -> int. Current file position)	
truncate(...)	Size defaults to the current file position, as returned by tell().The current file pos
writable(...)	
write(...)	Only makes one system call, so not all of the data may be written.
writelines	
close(() -> None. Close the file.)	A closed file cannot be used for further I/O operations.
fileno(() -> int. “file descriptor”.)	This is needed for lower-level file interfaces, such the fcntl module.
flush	Flush write buffers, if applicable.
isatty(...)	
read(...)	Only makes one system call, so less data may be returned than requested In non-
readable(...)	
readall(...)	In non-blocking mode, returns as much as is immediately available, or None if n
readline	Read and return a line from the stream.
readlines	Return a list of lines from the stream.
seek((offset: int[, ...)	Argument offset is a byte count.
seekable(...)	
tell(() -> int. Current file position)	
truncate(...)	Size defaults to the current file position, as returned by tell().The current file pos
writable(...)	
writelines	
close(() -> None. Close the file.)	A closed file cannot be used for further I/O operations.
fileno(() -> int. “file descriptor”.)	This is needed for lower-level file interfaces, such the fcntl module.
flush	Flush write buffers, if applicable.
isatty(...)	
readable(...)	
readline	Read and return a line from the stream.
readlines	Return a list of lines from the stream.
seek((offset: int[, ...)	Argument offset is a byte count.
seekable(...)	
tell(() -> int. Current file position)	
truncate(...)	Size defaults to the current file position, as returned by tell().The current file pos
writable(...)	
writelines	

Detailed Methods

close() → None. Close the file.
A closed file cannot be used for further I/O operations. close() may be called more than once without error. Changes the fileno to -1.

fileno() → int. “file descriptor”.

This is needed for lower-level file interfaces, such the fcntl module.

flush()
Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

isatty() → bool. True if the file is connected to a tty device.

read(*size: int*) → bytes. read at most size bytes, returned as bytes.
Only makes one system call, so less data may be returned than requested In non-blocking mode, returns None if no data is available. On end-of-file, returns ‘’.

readable() → bool. True if file was opened in a read mode.

readall() → bytes. read all data from the file, returned as bytes.
In non-blocking mode, returns as much as is immediately available, or None if no data is available. On end-of-file, returns ‘’.

readinto() → Same as RawIOBase.readinto().

readline()
Read and return a line from the stream.

If limit is specified, at most limit bytes will be read.

The line terminator is always b’n’ for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

readlines()
Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

seek(*offset: int*[, *whence: int*]) → None. Move to new file position.
Argument offset is a byte count. Optional argument whence defaults to 0 (offset from start of file, offset should be >= 0); other values are 1 (move relative to current position, positive or negative), and 2 (move relative to end of file, usually negative, although many platforms allow seeking beyond the end of a file). Note that not all file objects are seekable.

seekable() → bool. True if file supports random-access.

tell() → int. Current file position

truncate([*size: int*]) → None. Truncate the file to at most size bytes.
Size defaults to the current file position, as returned by tell().The current file position is changed to the value of size.

writable() → bool. True if file was opened in a write mode.

write(*b: bytes*) → int. Write bytes b to file, return number written.
Only makes one system call, so not all of the data may be written. The number of bytes actually written is returned.

writelines()

close() → None. Close the file.
A closed file cannot be used for further I/O operations. close() may be called more than once without error. Changes the fileno to -1.

fileno() → int. “file descriptor”.
This is needed for lower-level file interfaces, such the fcntl module.

flush()
Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

isatty() → bool. True if the file is connected to a tty device.

read(*size: int*) → bytes. read at most size bytes, returned as bytes.
Only makes one system call, so less data may be returned than requested In non-blocking mode, returns None if no data is available. On end-of-file, returns ‘’.

readable() → bool. True if file was opened in a read mode.

readall() → bytes. read all data from the file, returned as bytes.
In non-blocking mode, returns as much as is immediately available, or None if no data is available. On end-of-file, returns ‘’.

readline()
Read and return a line from the stream.

If limit is specified, at most limit bytes will be read.

The line terminator is always b’n’ for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

readlines()
Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

seek(*offset: int*[, *whence: int*]) → None. Move to new file position.
Argument offset is a byte count. Optional argument whence defaults to 0 (offset from start of file, offset should be >= 0); other values are 1 (move relative to current position, positive or negative), and 2 (move relative to end of file, usually negative, although many platforms allow seeking beyond the end of a file). Note that not all file objects are seekable.

seekable() → bool. True if file supports random-access.

tell() → int. Current file position

truncate([*size: int*]) → None. Truncate the file to at most size bytes.
Size defaults to the current file position, as returned by tell().The current file position is changed to the value of size.

writable() → bool. True if file was opened in a write mode.

writelines()

close() → None. Close the file.
A closed file cannot be used for further I/O operations. close() may be called more than once without error. Changes the fileno to -1.

fileno() → int. “file descriptor”.
This is needed for lower-level file interfaces, such the fcntl module.

flush()
Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

isatty() → bool. True if the file is connected to a tty device.

readable() → bool. True if file was opened in a read mode.

readline()
Read and return a line from the stream.

If limit is specified, at most limit bytes will be read.

The line terminator is always b'n' for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

readlines()
Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

seek(*offset: int*[, *whence: int*]) → None. Move to new file position.
Argument offset is a byte count. Optional argument whence defaults to 0 (offset from start of file, offset should be >= 0); other values are 1 (move relative to current position, positive or negative), and 2 (move relative to end of file, usually negative, although many platforms allow seeking beyond the end of a file). Note that not all file objects are seekable.

seekable() → bool. True if file supports random-access.

tell() → int. Current file position

truncate([*size: int*]) → None. Truncate the file to at most size bytes.
Size defaults to the current file position, as returned by tell().The current file position is changed to the value of size.

writable() → bool. True if file was opened in a write mode.

writelines()

spinnman.data.little_endian_byte_array_byte_reader module

class spinnman.data.little_endian_byte_array_byte_reader.**LittleEndianByteArrayByteReader**(*data*)
Bases: spinnman.data.abstract_byte_reader.AbstractByteReader

A byte reader that reads from a byte array using little endian notation

Parameters **data** (*bytearray*) – The byte array to read the data from

Methods

is_at_end()	ee spinnman.data.abstract_byte_reader.AbstractByteReader.is_at_end()
read_byte()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_byte()
read_bytes([size])	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_bytes()
read_int()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_int()
read_long()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_long()
read_short()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_short()

Detailed Methods

is_at_end()
ee spinnman.data.abstract_byte_reader.AbstractByteReader.is_at_end()

read_byte()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_byte()

read_bytes(*size=None*)

See spinnman.data.abstract_byte_reader.AbstractByteReader.read_bytes()

read_int()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_int()

read_long()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_long()

read_short()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_short()

spinnman.data.little_endian_byte_array_byte_writer module

class spinnman.data.little_endian_byte_array_byte_writer.**LittleEndianByteArrayByteWriter**
Bases: spinnman.data.abstract_byte_writer.AbstractByteWriter

A byte writer that writes to a byte array using little endian notation

Attributes

data	The data that was written

Methods

get_n_bytes_written()	See spinnman.data.abstract_byte_writer.AbstractByteWriter.get_n_byte
write_byte(byte_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_byte
write_bytes(byte_iterable)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_byte
write_int(int_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_int(
write_long(long_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_long
write_short(short_value)	See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_shor

Detailed Methods

get_n_bytes_written()
See spinnman.data.abstract_byte_writer.AbstractByteWriter.get_n_bytes_written()

write_byte(*byte_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_byte()

write_bytes(*byte_iterable*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_bytes()

write_int(*int_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_int()

write_long(*long_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_long()

write_short(*short_value*)
See spinnman.data.abstract_byte_writer.AbstractByteWriter.write_short()

spinnman.data.little_endian_data_reader_byte_reader module

class spinnman.data.little_endian_data_reader_byte_reader.**LittleEndianDataReaderByteReader**(*da*
Bases: spinnman.data.abstract_byte_reader.AbstractByteReader

A byte reader that reads from a data reader using little endian notation

Parameters **data_reader** (spinnman.data.abstract_data_reader.AbstractDataReader) – The data reader to read from

Methods

is_at_end()	required from abstract byte reader
read_byte()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_byte()
read_bytes([size])	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_bytes()
read_int()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_int()
read_long()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_long()
read_short()	See spinnman.data.abstract_byte_reader.AbstractByteReader.read_short()

Detailed Methods

is_at_end()
required from abstract byte reader

Returns

read_byte()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_byte()

read_bytes(*size=None*)
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_bytes()

read_int()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_int()

read_long()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_long()

read_short()
See spinnman.data.abstract_byte_reader.AbstractByteReader.read_short()

spinnman.messages package

Subpackages

spinnman.messages.eieio package

Subpackages

spinnman.messages.eieio.abstract_messages package

Submodules

spinnman.messages.eieio.abstract_messages.abstract_eieio_message module

class spinnman.messages.eieio.abstract_messages.abstract_eieio_message.**AbstractEIEIOMessage**

Bases: object

Marker interface for an EIEIOMessage

Abstract Methods

write_eieio_message(byte_writer)	Write the message to a byte writer

Detailed Methods

write_eieio_message(*byte_writer*)

Write the message to a byte writer

Parameters **byte_writer** (spinnman.data.abstract_byte_writer.AbstractByteWriter) – The writer to write to

spinnman.messages.eieio.command_messages package

Submodules

spinnman.messages.eieio.command_messages.database_confirmation module

class spinnman.messages.eieio.command_messages.database_confirmation.**DatabaseConfirmation**(*data*

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

data
database_path
eieio_header

Methods

read_eieio_command_message(command_header, ...)	
write_eieio_message(writer)	

Detailed Methods

static read_eieio_command_message(*command_header*, *byte_reader*)

write_eieio_message(*writer*)

spinnman.messages.eieio.command_messages.eieio_command_header module

class spinnman.messages.eieio.command_messages.eieio_command_header.**EIEIOCommandHeader**(*comman*

Bases: object

EIEIO header for command packets

Attributes

command

Methods

read_eieio_header(byte_reader)	Read an eieio command header from a byte_reader
write_eieio_header(writer)	Write the command header to a writer

Detailed Methods

static read_eieio_header (*byte_reader*)
Read an eieio command header from a byte_reader

Parameters **byte_reader** (spinnman.data.abstract_byte_reader.AbstractByteReader) – The reader to read the data from

Returns an eieio command header

Return type spinnman.messages.eieio.command_messages.eieio_command_header.EIEIOCom

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error reading from the reader
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If there is an error setting any of the values

write_eieio_header (*writer*)
Write the command header to a writer

Parameters **writer** (spinnman.data.abstract_byte_writer.AbstractByteWriter) – the writer to write the header to

Returns None

spinnman.messages.eieio.command_messages.eieio_command_message module

class spinnman.messages.eieio.command_messages.eieio_command_message.**EIEIOCommandMessage** (*eieio_ data_*

Bases: spinnman.messages.eieio.abstract_messages.abstract_eieio_message.AbstractEIEIOMess

An EIEIO command message

Parameters

- **eieio_command_header** (spinnman.messages.eieio.command_messages.eieio_command_head – The header of the message
- **data_reader** (spinnman.data.abstract_data_reader.AbstractDataReader) – Optional reader of incoming data

Attributes

	data
	eieio_header

Methods

	get_min_packet_length()
	read_eieio_command_message(command_header, ...)
	write_eieio_message(writer)

Detailed Methods

static get_min_packet_length()

static read_eieio_command_message(*command_header*, *byte_reader*)

write_eieio_message(*writer*)

spinnman.messages.eieio.command_messages.event_stop_request module

class spinnman.messages.eieio.command_messages.event_stop_request.**EventStopRequest**

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

	data
	eieio_header

spinnman.messages.eieio.command_messages.host_data_read module

class spinnman.messages.eieio.command_messages.host_data_read.**HostDataRead**(*region_id*, *sequence_no*, *space_read*)

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

	data
	eieio_header
	region_id
	sequence_no
	space_read

Methods

	get_min_packet_length()
	read_eieio_command_message(command_header, ...)
	write_eieio_message(writer)

Detailed Methods

static get_min_packet_length()

static read_eieio_command_message(*command_header*, *byte_reader*)

write_eieio_message(*writer*)

spinnman.messages.eieio.command_messages.host_send_sequenced_data module

class spinnman.messages.eieio.command_messages.host_send_sequenced_data.**HostSendSequencedData**

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

data
eieio_data_message
eieio_header
region_id
sequence_no

Methods

get_min_packet_length()
read_eieio_command_message(command_header, ...)
write_eieio_message(writer)

Detailed Methods

static get_min_packet_length()

static read_eieio_command_message(*command_header*, *byte_reader*)

write_eieio_message(*writer*)

spinnman.messages.eieio.command_messages.padding_request module

class spinnman.messages.eieio.command_messages.padding_request.**PaddingRequest**

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

data
eieio_header

Methods

	get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.command_messages.spinnaker_request_buffers module

class spinnman.messages.eieio.command_messages.spinnaker_request_buffers.**SpinnakerRequestBuff**

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

	data
	eieio_header
	p
	region_id
	sequence_no
	space_available
	x
	y

Methods

	get_min_packet_length()
	read_eieio_command_message(command_header, ...)
	write_eieio_message(writer)

Detailed Methods

static get_min_packet_length()

static read_eieio_command_message(*command_header*, *byte_reader*)

write_eieio_message(*writer*)

spinnman.messages.eieio.command_messages.spinnaker_request_read_data module

class spinnman.messages.eieio.command_messages.spinnaker_request_read_data.**SpinnakerRequestRe**

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

data
eieio_header
p
region_id
sequence_no
space_available
x
y

Methods

get_min_packet_length()
read_eieio_command_message(command_header, ...)
write_eieio_message(writer)

Detailed Methods

static get_min_packet_length()

static read_eieio_command_message(*command_header*, *byte_reader*)

write_eieio_message(*writer*)

spinnman.messages.eieio.command_messages.start_requests module

class spinnman.messages.eieio.command_messages.start_requests.**StartRequests**

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

data
eieio_header

spinnman.messages.eieio.command_messages.stop_requests module

class spinnman.messages.eieio.command_messages.stop_requests.**StopRequests**

Bases: spinnman.messages.eieio.command_messages.eieio_command_message.EIEIOCommandMessage

Attributes

data	
eieio_header	

spinnman.messages.eieio.data_messages package

Subpackages

spinnman.messages.eieio.data_messages.eieio_16bit package

Submodules

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_data_message.**EIEIO16BitDa**

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()	

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_lower_key_prefix_data_mes

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header

Continued on next page

Table 3.94 – continued from previous page

is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_payload_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_payload_prefix_data_messa

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_payload_prefix_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_payload_prefix_lower_key_

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_payload_prefix_upper_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_payload_prefix_upper_key_

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_timed_payload_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_timed_payload_prefix_data

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_timed_payload_prefix_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_timed_payload_prefix_lowe

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_timed_payload_prefix_upper_key_prefix_data_message module
class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_timed_payload_prefix_upper

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_upper_key_prefix_data_message module
class spinnman.messages.eieio.data_messages.eieio_16bit.eieio_16bit_upper_key_prefix_data_mes

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload package

Submodules

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payloac

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payloac

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header

Continued on next page

Table 3.112 – continued from previous page

is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_payload_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payloac

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_payload_prefix_lower_key_prefix_ module

class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payloac

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_payload_prefix_upper_key_prefix module

class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_timed_data_message module

class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_timed_lower_key_prefix_data_me
module
class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_timed_upper_key_prefix_data_me
module
class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload_upper_key_prefix_data_message
module
class spinnman.messages.eieio.data_messages.eieio_16bit_with_payload.eieio_16bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit package

Submodules

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_data_message.**EIEIO32BitDa**

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_lower_key_prefix_data_mes

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_payload_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_payload_prefix_data_messa

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_payload_prefix_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_payload_prefix_lower_key_

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet

Continued on next page

Table 3.134 – continued from previous page

n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_payload_prefix_upper_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_payload_prefix_upper_key_

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_timed_payload_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_timed_payload_prefix_data

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_timed_payload_prefix_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_timed_payload_prefix_lowe

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWith

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_timed_payload_prefix_upper_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_timed_payload_prefix_upper_key_data_message.EIEIO32BitTimedPayloadPrefixUpperKeyDataMessage(*eieio_header=None*, *data=None*, *offset=0*)

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWithoutPayloadDataMessage

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_upper_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit.eieio_32bit_upper_key_prefix_data_message.EIEIO32BitUpperKeyPrefixDataMessage(*eieio_header=None*, *data=None*, *offset=0*)

Bases: spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.EIEIOWithoutPayloadDataMessage

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload package

Submodules

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_lower_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_payload_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_payload_prefix_lower_key_prefix module

class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet

Continued on next page

Table 3.152 – continued from previous page

next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_payload_prefix_upper_key_prefix module

class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payloac

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_timed_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payloac

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_timed_lower_key_prefix_data_me
module
class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payloa

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPa

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_timed_upper_key_prefix_data_m
module
class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payloa

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPa

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload_upper_key_prefix_data_message module

class spinnman.messages.eieio.data_messages.eieio_32bit_with_payload.eieio_32bit_with_payload

Bases: spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.EIEIOWithPay

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

get_min_packet_length()

Detailed Methods

static get_min_packet_length()

Submodules

spinnman.messages.eieio.data_messages.abstract_eieio_data_element module

class spinnman.messages.eieio.data_messages.abstract_eieio_data_element.**AbstractEIEIODataElem**
Bases: object

A marker interface for possible data elements in the EIEIO data packet

Abstract Methods

write_element(eieio_type, byte_writer)	Write the element to the writer given the type

Detailed Methods

write_element(*eieio_type*, *byte_writer*)
Write the element to the writer given the type

Parameters

- **eieio_type** (spinnman.messages.eieio.eieio_type.EIEIOType) – The type of the message being written
- **byte_writer** (spinnman.data.abstract_byte_writer.AbstractByteWriter) – The writer to write to

Raises SpinnmanInvalidParameterException If the type is incompatible with the element

spinnman.messages.eieio.data_messages.eieio_data_header module

class spinnman.messages.eieio.data_messages.eieio_data_header.**EIEIODataHeader**(*eieio_type*, *tag=0*, *prefix=None*, *prefix_type=<EIEIOPref 0>*, *payload_base=None*, *is_time=False*, *count=0*)
Bases: object

EIEIO header for data packets

Parameters

- **eieio_type** (spinnman.spinnman.messages.eieio.eieio_type.EIEIOType) – the type of message
- **tag** (*int*) – the tag of the message (0 by default)
- **prefix** (*int or None*) – the key prefix of the message or None if not prefixed
- **prefix_type** (spinnman.messages.eieio.eieio_prefix.EIEIOPrefix) – the position of the prefix (upper or lower)
- **payload_base** (*int or None*) – The base payload to be applied, or None if no base payload
- **is_time** (*bool*) – True if the payloads should be taken to be timestamps, or False otherwise
- **count** (*int*) – Count of the number of items in the packet

Attributes

count
eieio_type
is_time
payload_base
prefix
prefix_type
size
tag

Methods

get_header_size(eieio_type[, is_prefix, ...])	Get the size of a header with the given parameters
increment_count()	
read_eieio_header(byte_reader)	Read an eieio data header from a byte_reader
reset_count()	
write_eieio_header(byte_writer)	Writes the header to a writer

Detailed Methods

static get_header_size(*eieio_type*, *is_prefix=False*, *is_payload_base=False*)
Get the size of a header with the given parameters

Parameters

- **eieio_type** (spinnman.spinnman.messages.eieio.eieio_type.EIEIOType) – the type of message
- **is_prefix** (*bool*) – True if there is a prefix, False otherwise
- **is_payload_base** (*bool*) – True if there is a payload base, False otherwise

Returns The size of the header in bytes

Return type int

increment_count()

static read_eieio_header(*byte_reader*)
Read an eieio data header from a byte_reader

Parameters **byte_reader** (spinnman.data.abstract_byte_reader.AbstractByteReader) – The reader to read the data from

Returns an eieio header

Return type spinnman.messages.eieio.data_messages.eieio_data_header.EIEIODataHeade

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error reading from the reader
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If there are too few bytes to read the header

- **spinnman.exceptions.SpinnmanInvalidParameterException** – If there is an error setting any of the values

reset_count()

write_eieio_header(*byte_writer*)
Writes the header to a writer

Parameters **byte_writer** (spinnman.data.abstract_byte_writer.AbstractByteWriter) – The writer to write the header to

spinnman.messages.eieio.data_messages.eieio_data_message module

class spinnman.messages.eieio.data_messages.eieio_data_message.**EIEIODataMessage**(*eieio_header*, *data_reader=None*

Bases: spinnman.messages.eieio.abstract_messages.abstract_eieio_message.AbstractEIEIOMess

An EIEIO Data message

Parameters

- **eieio_header** (spinnman.messages.eieio.data_messages.eieio_data_header.EIEIODataHe – The header of the message
- **data_reader** (spinnman.data.abstract_data_reader.AbstractDataReader) – Optional reader of data contained within the packet, or None if this packet is being written

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

add_element(element)	Add an element to the message.
min_packet_length(eieio_type[, is_prefix, ...])	The minimum length of a message with the given header, in bytes
write_eieio_message(byte_writer)	

Detailed Methods

add_element(*element*)
Add an element to the message. The correct type of element must be added, depending on the header values

Parameters **element** (spinnman.messages.eieio.data_messages.abstract_eieio_data_elemen – The element to be added

Raises

- **SpinnmanInvalidParameterException** – If the element is not compatible with the header

- **SpinnmanInvalidPacketException** – If the message was created to read data from a reader

static **min_packet_length**(*eieio_type*, *is_prefix=False*, *is_payload_base=False*)
The minimum length of a message with the given header, in bytes

Parameters

- **eieio_type** (spinnman.spinnman.messages.eieio.eieio_type.EIEIOType) – the type of message
- **is_prefix** (*bool*) – True if there is a prefix, False otherwise
- **is_payload_base** (*bool*) – True if there is a payload base, False otherwise

Returns The minimum size of the packet in bytes

Return type int

write_eieio_message(*byte_writer*)

spinnman.messages.eieio.data_messages.eieio_key_data_element module

class spinnman.messages.eieio.data_messages.eieio_key_data_element.**EIEIOKeyDataElement**(*key*)
Bases: spinnman.messages.eieio.data_messages.abstract_eieio_data_element.AbstractEIEIODat

A data element that contains just a key

Attributes

key

Methods

write_element(eieio_type, byte_writer)

Detailed Methods

write_element(*eieio_type*, *byte_writer*)

spinnman.messages.eieio.data_messages.eieio_key_payload_data_element module

class spinnman.messages.eieio.data_messages.eieio_key_payload_data_element.**EIEIOKeyPayloadDat**

Bases: spinnman.messages.eieio.data_messages.abstract_eieio_data_element.AbstractEIEIODat

A data element that contains a key and a payload

Attributes

key

Continued on next page

Table 3.171 – continued from previous page

payload
payload_is_timestamp

Methods

write_element(eieio_type, byte_writer)	

Detailed Methods

write_element(*eieio_type*, *byte_writer*)

spinnman.messages.eieio.data_messages.eieio_with_payload_data_message module

class spinnman.messages.eieio.data_messages.eieio_with_payload_data_message.**EIEIOWithPayloadD**

Bases: spinnman.messages.eieio.data_messages.eieio_data_message.EIEIODataMessage

An EIEIO message with a payload

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

add_key_and_payload(key, payload)	Adds a key and payload to the packet

Detailed Methods

add_key_and_payload(*key*, *payload*)
Adds a key and payload to the packet

Parameters

- **key** (*int*) – The key to add
- **payload** (*int*) – The payload to add

Raises SpinnmanInvalidParameterException If the key or payload is too big for the format

spinnman.messages.eieio.data_messages.eieio_without_payload_data_message module

class spinnman.messages.eieio.data_messages.eieio_without_payload_data_message.**EIEIOWithoutPa**

Bases: spinnman.messages.eieio.data_messages.eieio_data_message.EIEIODataMessage

An EIEIO message without a payload

Attributes

eieio_header	
is_next_element	Determine if there is another element to be read
max_n_elements	The maximum number of elements that can fit in the packet
n_elements	The number of elements in the packet
next_element	The next element to be read, or None if no more elements.
size	The size of the packet with the current contents

Methods

add_key(key)	Add a key to the packet

Detailed Methods

add_key (*key*)
Add a key to the packet

Parameters **key** (*int*) – The key to add

Raises **SpinnmanInvalidParameterException** If the key is too big for the format

Submodules

spinnman.messages.eieio.create_eieio_command module

Functions

read_eieio_command_message(byte_reader)	

spinnman.messages.eieio.create_eieio_command.**read_eieio_command_message**(*byte_reader*)

spinnman.messages.eieio.create_eieio_data module

Functions

read_eieio_data_message(byte_reader)	

spinnman.messages.eieio.create_eieio_data.**read_eieio_data_message**(*byte_reader*)

spinnman.messages.eieio.eieio_prefix module

class spinnman.messages.eieio.eieio_prefix.**EIEIOPrefix**(*value*, *doc=''*)
Bases: enum.Enum

Possible prefixing of keys in EIEIO packets

Attributes

LOWER_HALF_WORD	apply prefix on lower half of the word
UPPER_HALF_WORD	apply prefix on top half of the word

spinnman.messages.eieio.eieio_type module

class spinnman.messages.eieio.eieio_type.**EIEIOType**(*value*, *key_bytes*, *payload_bytes*, *doc=''*)
Bases: enum.Enum

Possible types of EIEIO packets

Attributes

KEY_16_BIT	Indicates that data is keys which are 16 bits
KEY_32_BIT	Indicates that data is keys of 32 bits
KEY_PAYLOAD_16_BIT	Indicates that data is keys and payloads of 16 bits
KEY_PAYLOAD_32_BIT	Indicates that data is keys and payloads of 32 bits

spinnman.messages.scp package

Subpackages

spinnman.messages.scp.abstract_messages package

Submodules

spinnman.messages.scp.abstract_messages.abstract_scp_request module

class spinnman.messages.scp.abstract_messages.abstract_scp_request.**AbstractSCPRequest**(*sdp_head*
scp_reque
ar-
gu-
ment_1=N
ar-
gu-
ment_2=N
ar-
gu-
ment_3=N
data=Non

Bases: object

Represents an Abstract SCP Request

Parameters

- **sdp_header** (spinnman.messages.sdp.sdp_header.SDPHeader) – The SDP header of the request
- **scp_request_header** (spinnman.messages.scp.scp_request_header.SCPRequestHeader) – The SCP header of the request
- **argument_1** (*int*) – The first argument, or None if no first argument
- **argument_2** (*int*) – The second argument, or None if no second argument
- **argument_3** (*int*) – The third argument, or None if no third argument
- **data** (*bytearray*) – The optional data, or None if no data

Raises None No known exceptions are raised

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Abstract Methods

get_scp_response()	Get an SCP response message to be used to process any response received

Methods

write_scp_request(byte_writer)	Write the scp request to the given writer

Detailed Methods

get_scp_response()

Get an SCP response message to be used to process any response received

Returns An SCP response, or None if no response is required

Return type spinnman.messages.scp_response.SCPResponse

Raises None No known exceptions are raised

write_scp_request(*byte_writer*)

Write the scp request to the given writer

Parameters **byte_writer** (spinnman.data.abstract_byte_writer.AbstractByteWriter) – The writer to write to

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error writing the request
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If any required values have not been set

spinnman.messages.scp.abstract_messages.abstract_scp_response module

class spinnman.messages.scp.abstract_messages.abstract_scp_response.**AbstractSCPResponse**

Bases: object

Represents an abstract SCP Response

Attributes

scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response

Abstract Methods

read_scp_response(byte_reader)	Read the scp response from the given reader

Detailed Methods

read_scp_response(*byte_reader*)

Read the scp response from the given reader

Parameters **byte_reader** (spinnman.data.abstract_byte_reader.AbstractByteReader) – The reader to read from

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error reading from the reader
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If there are not enough bytes to read the header
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If there is an error setting any of the values
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If the response code indicates an error

spinnman.messages.scp.impl package

Submodules

spinnman.messages.scp.impl.scp_app_stop_request module

class spinnman.messages.scp.impl.scp_app_stop_request.**SCPAppStopRequest**(*app_id*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to stop an application

Parameters

- **app_id** (*int*) – The id of the application, between 0 and 255
- **signal** (spinnman.messages.scp.scp_signal.SCPSignal) – The signal to send

Raises spinnman.exceptions.SpinnmanInvalidParameterException If app_id is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_application_run_request module

class spinnman.messages.scp.impl.scp_application_run_request.**SCPApplicationRunRequest**(*app_id*, *x*, *y*, *processors*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP request to run an application loaded on a chip

Parameters

- **app_id** (*int*) – The id of the application to run, between 16 and 255
- **x** (*int*) – The x-coordinate of the chip to run on, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip to run on, between 0 and 255
- **processors** – The processors on the chip where the executable should be started, between 1 and 17

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the app_id is out of range

- If the chip coordinates are out of range
- If one of the processors is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()	See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_sc

Detailed Methods

get_scp_response()
See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_scp_response()

spinnman.messages.scp.impl.scp_check_ok_response module

class spinnman.messages.scp.impl.scp_check_ok_response.**SCPCheckOKResponse**(*operation*, *command*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request which returns nothing other than OK

Parameters

- **operation** (*str*) – The operation being performed
- **command** (*str*) – The command that was sent

Attributes

scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)
See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_count_state_request module

class spinnman.messages.scp.impl.scp_count_state_request.**SCPCountStateRequest**(*app_id*, *state*)
Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to get a count of the cores in a particular state

Parameters

- **app_id** (*int*) – The id of the application, between 0 and 255
- **state** (spinnman.model.cpu_state.CPUState) – The state to count

Raises spinnman.exceptions.SpinnmanInvalidParameterException If app_id is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_count_state_response module

class spinnman.messages.scp.impl.scp_count_state_response.**SCPCountStateResponse**
Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request for the number of cores in a given state

Attributes

count	The count of the number of cores with the requested state
scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)
See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_flood_fill_data_request module

class spinnman.messages.scp.impl.scp_flood_fill_data_request.**SCPFloodFillDataRequest**(*nearest_nei*
block_no,
base_addre
data)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to start a flood fill of data

Parameters

- **nearest_neighbour_id** (*int*) – The id of the packet, between 0 and 127
- **block_no** (*int*) – Which block this block is, between 0 and 255
- **base_address** (*int*) – The base address where the data is to be loaded
- **data** (*bytearray*) – The data to load, between 4 and 256 bytes and the size must be divisible by 4

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the id is out of range
- If the block number is out of range
- If the base_address is not a positive integer
- If the data is too long or too short
- If the length of the data is not divisible by 4

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_flood_fill_end_request module

class spinnman.messages.scp.impl.scp_flood_fill_end_request.**SCPFloodFillEndRequest**(*nearest_neighbour_id*, *app_id=0*, *processors=None*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to start a flood fill of data

Parameters

- **nearest_neighbour_id** (*int*) – The id of the packet, between 0 and 127
- **app_id** (*int*) – The application id to start using the data, between 16 and 255. If not specified, no application is started
- **processors** (*iterable of int*) – A list of processors on which to start the application, each between 1 and 17. If not specified, no application is started.

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the id is out of range
- If the app_id is out of range
- If one of the processors is out of range
- If only one of app_id or processors is specified

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_flood_fill_start_request module

class spinnman.messages.scp.impl.scp_flood_fill_start_request.**SCPFloodFillStartRequest**(*nearest_*
n_blocks
x=None
y=None

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to start a flood fill of data

Parameters

- **nearest_neighbour_id** (*int*) – The id of the packet, between 0 and 127
- **n_blocks** (*int*) – The number of blocks of data that will be sent, between 0 and 255
- **x** (*int*) – The x-coordindate of the chip to load the data on to. If not specified, the data will be loaded on to all chips
- **y** (*int*) – The y-coordinate of the chip to load the data on to. If not specified, the data will be loaded on to all chips

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the id is out of range
- If the number of blocks is out of range
- If only one of x or y is specified
- If x and y are out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_iptag_clear_request module

class spinnman.messages.scp.impl.scp_iptag_clear_request.**SCPIPTagClearRequest**(*x*,
y,
tag)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to clear an IP Tag

Parameters

- **x** (*int*) – The x-coordinate of a chip, between 0 and 255
- **y** (*int*) – The y-coordinate of a chip, between 0 and 255
- **tag** (*int*) – The tag, between 0 and 7

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- The chip-coordinates are out of range
- If the tag is not between 0 and 7

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_iptag_get_request module

class spinnman.messages.scp.impl.scp_iptag_get_request.**SCPTagGetRequest**(*x*, *y*, *tag*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to get an IP tag

Parameters

- **x** (*int*) – The x-coordinate of a chip, between 0 and 255
- **y** (*int*) – The y-coordinate of a chip, between 0 and 255
- **tag** (*int*) – The tag to get details of, between 0 and 7
- **tag** – The tag, between 0 and 7

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- The chip-coordinates are out of range
- If the tag is not between 0 and 7

Attributes

Continued on next page

Table 3.204 – continued from previous page

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_iptag_get_response module

class spinnman.messages.scp.impl.scp_iptag_get_response.**SCPIPTagGetResponse**

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request for an IP tags

Attributes

count	The count of the number of packets that have been sent with the tag
flags	The flags of the tag
in_use	True if the tag is marked as being in use
ip_address	The IP address of the tag, as a bytearray of 4 bytes
is_arp	True if the tag is in the ARP state (where the MAC address is being looked up - transient state so unlik
is_reverse	True if the tag is a reverse tag
is_temporary	True if the tag is temporary
mac_address	The MAC address of the tag, as a bytearray of 6 bytes
port	The port of the tag
rx_port	The receive port of the tag
scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response
spin_chip_x	The x-coordinate of the chip on which the tag is defined
spin_chip_y	The y-coordinate of the chip on which the tag is defined
spin_cpu	The cpu id of the ip tag
spin_port	The spin-port of the ip tag
strip_sdp	True if the tag is to strip sdp
timeout	The timeout of the tag

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)
See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_iptag_info_request module

class spinnman.messages.scp.impl.scp_iptag_info_request.**SCPTagInfoRequest**(*x*, *y*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request information about IP tags

Parameters

- **x** (*int*) – The x-coordinate of a chip, between 0 and 255
- **y** (*int*) – The y-coordinate of a chip, between 0 and 255

Raises spinnman.exceptions.SpinnmanInvalidParameterException The chip-coordinates are out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_iptag_info_response module

class spinnman.messages.scp.impl.scp_iptag_info_response.**SCPIPTagInfoResponse**

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request for information about IP tags

Attributes

fixed_size	The count of the number of fixed IP tag entries
pool_size	The count of the IP tag pool size
scp_response_header	The SCP header from the response

Continued on next page

Table 3.210 – continued from previous page

sdp_header	The SDP header from the response

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)
See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_iptag_set_request module

class spinnman.messages.scp.impl.scp_iptag_set_request.**SCPIPTagSetRequest**(*x*, *y*, *host*, *port*, *tag*, *strip*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to set an IP Tag

Parameters

- **x** (*int*) – The x-coordinate of a chip, between 0 and 255
- **y** (*int*) – The y-coordinate of a chip, between 0 and 255
- **host** (*bytearray*) – The host address, as an array of 4 bytes
- **port** (*int*) – The port, between 0 and 65535
- **tag** (*int*) – The tag, between 0 and 7
- **strip** (*bool*) – if the SDP header should be striped from the packet.

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- The chip-coordinates are out of range
- If the host is not 4 bytes
- If the port is not between 0 and 65535
- If the tag is not between 0 and 7

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()	

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_led_request module

class spinnman.messages.scp.impl.scp_led_request.**SCPLEDRequest**(*x*, *y*, *cpu*, *led_states*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to change the state of an LED

Parameters

- **x** (*int*) – The x-coordinate of the chip, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip, between 0 and 255
- **cpu** (*int*) – The CPU-number to use to set the LED.
- **led_states** (*dict*) – A dictionary mapping LED index to state with 0 being off, 1 on and 2 inverted.

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If x is out of range
- If y is out of range
- If cpu is out of range
- If LEDs are not in the range 0 to 7
- If LED states are not 0, 1 or 2.

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()	See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_sc

Detailed Methods

get_scp_response()
See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_scp_response()

spinnman.messages.scp.impl.scp_read_link_request module

class spinnman.messages.scp.impl.scp_read_link_request.**SCPReadLinkRequest**(*x*, *y*, *cpu*, *link*, *base_address*, *size*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP request to read a region of memory via a link on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip to read from, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip to read from, between 0 and 255
- **cpu** (*int*) – The CPU core to use, normally 0 (or if a BMP, the board slot number)
- **link** (*int*) – The id of the link down which to send the query
- **base_address** (*int*) – The positive base address to start the read from
- **size** (*int*) – The number of bytes to read, between 1 and 256

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the chip coordinates are out of range
- If the base address is not a positive number
- If the size is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()	See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_sc

Detailed Methods

get_scp_response()
See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_scp_response()

spinnman.messages.scp.impl.scp_read_link_response module

class spinnman.messages.scp.impl.scp_read_link_response.**SCPReadLinkResponse**

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request to read a region of memory via a link on a chip

Attributes

data	The data read
scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)

See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_read_memory_request module

class spinnman.messages.scp.impl.scp_read_memory_request.**SCPReadMemoryRequest**(*x*, *y*, *base_address*, *size*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP request to read a region of memory on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip to read from, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip to read from, between 0 and 255
- **base_address** (*int*) – The positive base address to start the read from
- **size** (*int*) – The number of bytes to read, between 1 and 256

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the chip coordinates are out of range
- If the base address is not a positive number
- If the size is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument

Continued on next page

Table 3.220 – continued from previous page

argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()	See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_sc

Detailed Methods

get_scp_response()
See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_scp_response()

spinnman.messages.scp.impl.scp_read_memory_response module

class spinnman.messages.scp.impl.scp_read_memory_response.**SCPReadMemoryResponse**
Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request to read a region of memory on a chip

Attributes

data	The data read
scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)
See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_read_memory_words_request module

class spinnman.messages.scp.impl.scp_read_memory_words_request.**SCPReadMemoryWordsRequest**(*x*, *y*, *base_*
size)
Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP request to read a region of memory on a chip in words

Parameters

- **x** (*int*) – The x-coordinate of the chip to read from, between 0 and 255

- **y** (*int*) – The y-coordinate of the chip to read from, between 0 and 255
- **base_address** (*int*) – The positive base address to start the read from
- **size** (*int*) – The number of words to read, between 1 and 64

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the chip coordinates are out of range
- If the base address is not a positive number
- If the size is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()	See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_sc

Detailed Methods

get_scp_response()
See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_scp_response()

spinnman.messages.scp.impl.scp_read_memory_words_response module

class spinnman.messages.scp.impl.scp_read_memory_words_response.**SCPReadMemoryWordsResponse**
Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request to read a region of memory on a chip in words

Attributes

data	The words of data read
scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)
See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_reverse_iptag_set_request module

class spinnman.messages.scp.impl.scp_reverse_iptag_set_request.**SCPReverseIPTagSetRequest**(*x*, *y*, *destination_x*, *destination_y*, *destination_p*, *port*, *tag*, *sdp_p*

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to set an IP Tag

Parameters

- **x** (*int*) – The x-coordinate of a chip, between 0 and 255
- **y** (*int*) – The y-coordinate of a chip, between 0 and 255
- **destination_x** (*int*) – The x-coordinate of the destination chip, between 0 and 255
- **destination_y** (*int*) – The y-coordinate of the destination chip, between 0 and 255
- **destination_p** (*int*) – the id of the destination processor, between 0 and 17
- **port** (*int*) – The port, between 0 and 65535
- **tag** (*int*) – The tag, between 0 and 7

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- The chip-coordinates are out of range
- If the host is not 4 bytes
- If the port is not between 0 and 65535
- If the tag is not between 0 and 7

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument

Continued on next page

Table 3.228 – continued from previous page

data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_router_alloc_request module

class spinnman.messages.scp.impl.scp_router_alloc_request.**SCPRouterAllocRequest**(*x*, *y*, *app_id*, *n_entries*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to allocate space for routing entries

Parameters

- **x** (*int*) – The x-coordinate of the chip to allocate on, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip to allocate on, between 0 and 255
- **app_id** (*int*) – The id of the application, between 0 and 255
- **n_entries** (*int*) – The number of entries to allocate

Raises spinnman.exceptions.SpinnmanInvalidParameterException If app_id is out of range, or n_entries is 0 or less

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_router_alloc_response module

class spinnman.messages.scp.impl.scp_router_alloc_response.**SCPRouterAllocResponse**

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request to allocate router entries

Attributes

base_address	The base address allocated, or 0 if none
scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)

See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_router_clear_request module

class spinnman.messages.scp.impl.scp_router_clear_request.**SCPRouterClearRequest**(*x*, *y*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to clear the router on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip, between 0 and 255

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If x is out of range
- If y is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_router_init_request module

class spinnman.messages.scp.impl.scp_router_init_request.**SCPRouterInitRequest**(*x*, *y*, *n_entries*, *table_address*, *base_address*, *app_id*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to initialize the router on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip, between 0 and 255
- **n_entries** (*int*) – The number of entries in the table, more than 0
- **table_address** (*int*) – The allocated table address
- **base_address** (*int*) – The base_address containing the entries
- **app_id** (*int*) – The id of the application with which to associate the routes. If not specified, defaults to 0.

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If x is out of range
- If y is out of range
- If n_entries is 0 or less
- If table_address is not positive
- If base_address is not positive

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_send_signal_request module

class spinnman.messages.scp.impl.scp_send_signal_request.**SCPSendSignalRequest**(*app_id*, *signal*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP Request to send a signal to cores

Parameters

- **app_id** (*int*) – The id of the application, between 0 and 255
- **signal** (spinnman.messages.scp.scp_signal.SCPSignal) – The signal to send

Raises spinnman.exceptions.SpinnmanInvalidParameterException If app_id is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_version_request module

class spinnman.messages.scp.impl.scp_version_request.**SCPVersionRequest**(*x*, *y*, *p*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

An SCP request to read the version of software running on a core

Parameters

- **x** (*int*) – The x-coordinate of the chip to read from, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip to read from, between 0 and 255

- **p** (*int*) – The id of the processor to read the version from, between 0 and 31

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If the chip coordinates are out of range
- If the processor is out of range

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()	See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_sc

Detailed Methods

get_scp_response()
See spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest.get_scp_response()

spinnman.messages.scp.impl.scp_version_response module

class spinnman.messages.scp.impl.scp_version_response.**SCPVersionResponse**
Bases: spinnman.messages.scp.abstract_messages.abstract_scp_response.AbstractSCPResponse

An SCP response to a request for the version of software running

Attributes

scp_response_header	The SCP header from the response
sdp_header	The SDP header from the response
version_info	The version information received

Methods

read_scp_response(byte_reader)	See spinnman.messages.scp.abstract_scp_response.AbstractSCPResp

Detailed Methods

read_scp_response(*byte_reader*)
See spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_scp_respons

spinnman.messages.scp.impl.scp_write_link_request module

class spinnman.messages.scp.impl.scp_write_link_request.**SCPWriteLinkRequest**(*x*, *y*, *cpu*, *link*, *base_address*, *data*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to write memory on a neighbouring chip

Parameters

- **x** (*int*) – The x-coordinate of the chip whose neighbour will be written to, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip whose neighbour will be written to, between 0 and 255
- **cpu** (*int*) – The CPU core to use, normally 0 (or if a BMP, the board slot number)
- **link** (*int*) – The link number to write to between 0 and 5 (or if a BMP, the FPGA between 0 and 2)
- **base_address** (*int*) – The base_address to start writing to
- **data** (*bytearray*) – Up to 256 bytes of data to write

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If x is out of range
- If y is out of range
- If base_address is not positive
- If the length of data is 0 or more than 256

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_write_memory_request module

class spinnman.messages.scp.impl.scp_write_memory_request.**SCPWriteMemoryRequest**(*x*, *y*, *base_address*, *data*)

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to write memory on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip, between 0 and 255
- **y** (*int*) – The y-coordinate of the chip, between 0 and 255
- **base_address** (*int*) – The base_address to start writing to
- **data** (*bytearray*) – Up to 256 bytes of data to write

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If x is out of range
- If y is out of range
- If base_address is not positive
- If the length of data is 0 or more than 256

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

spinnman.messages.scp.impl.scp_write_memory_words_request module

class spinnman.messages.scp.impl.scp_write_memory_words_request.**SCPWriteMemoryWordsRequest**(*x*, *y*, *ba*, *da*

Bases: spinnman.messages.scp.abstract_messages.abstract_scp_request.AbstractSCPRequest

A request to write memory on a chip using words (little endian)

Parameters

- **x** (*int*) – The x-coordinate of the chip, between 0 and 255

- **y** (*int*) – The y-coordinate of the chip, between 0 and 255
- **base_address** (*int*) – The base_address to start writing to
- **data** (*iterable of int*) – Up to 64 ints of data to write

Raises spinnman.exceptions.SpinnmanInvalidParameterException

- If x is out of range
- If y is out of range
- If base_address is not positive
- If the length of data is 0 or more than 256

Attributes

argument_1	The first argument, or None if no first argument
argument_2	The second argument, or None if no second argument
argument_3	The third argument, or None if no third argument
data	The data, or None if no data
scp_request_header	The SCP request header of the message
sdp_header	The SDP header of the message

Methods

get_scp_response()

Detailed Methods

get_scp_response()

Submodules

spinnman.messages.scp.scp_command module

class spinnman.messages.scp.scp_command.**SCPCommand**(*value*, *doc=''*)

Bases: enum.Enum

The SCP Commands

Attributes

CMD_ALLOC	Router allocation
CMD_APLX	
CMD_AR	
CMD_AS	
CMD_FFD	Send Flood-Fill Data
CMD_FILL	
CMD_FLASH_COPY	

Continued on next page

Table 3.250 – continued from previous page

CMD_FLASH_ERASE	
CMD_FLASH_WRITE	
CMD_IPTAG	Set an IPTAG
CMD_LED	Control the LEDs
CMD_LINK_READ	Read neighbouring chip's memory.
CMD_LINK_WRITE	Write neighbouring chip's memory.
CMD_NNP	Send a Nearest-Neighbour packet
CMD_P2PC	
CMD_POWER	
CMD_READ	Read SDRAM
CMD_REMAP	
CMD_RESET	
CMD_RTR	Router initialization
CMD_RUN	
CMD_SIG	Send a Signal
CMD_SROM	
CMD_TUBE	
CMD_VER	Get SCAMP Version
CMD_WRITE	Write SDRAM

spinnman.messages.scp.scp_iptag_command module

class spinnman.messages.scp.scp_iptag_command.**SCPIPTagCommand**(*value*, *doc=''*)

Bases: enum.Enum

SCP IPTag Commands

Attributes

CLR
GET
NEW
SET
TTO

spinnman.messages.scp.scp_request_header module

class spinnman.messages.scp.scp_request_header.**SCPRequestHeader**(*command*, *sequence=None*)

Bases: object

Represents the header of an SCP Request Each optional parameter in the constructor can be set to a value other than None once, after which it is immutable. It is an error to set a parameter that is not currently None.

Parameters

- **command** (spinnman.messages.scp.scp_command.SCPCommand) – The SCP command
- **sequence** (*int*) – The number of the SCP packet in order of all packets sent or received, between 0 and 65535

Raises spinnman.exceptions.SpinnmanInvalidParameterException If one of the parameters is incorrect

Attributes

command	The command of the SCP packet
sequence	The sequence number of the SCP packet

Methods

write_scp_request_header(byte_writer)	Write the SCP header to a byte_writer

Detailed Methods

write_scp_request_header(*byte_writer*)
Write the SCP header to a byte_writer

Parameters **byte_writer** (spinnman.data.abstract_byte_writer.AbstractByteWriter) – The writer to write the data to

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error writing to the writer
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If any of the parameter values have not been set

spinnman.messages.scp.scp_response_header module

class spinnman.messages.scp.scp_response_header.**SCPResponseHeader**
Bases: object

Represents the header of an SCP Response

Attributes

result	The result of the SCP response
sequence	The sequence number of the SCP response

Methods

read_scp_response_header(byte_reader)	Read an SCP header from a byte_reader

Detailed Methods

read_scp_response_header(*byte_reader*)
Read an SCP header from a byte_reader

Parameters **byte_reader** (spinnman.data.abstract_byte_reader.AbstractByteReader) – The reader to read the data from

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error reading from the reader
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If there are not enough bytes to read the header
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If there is an error setting any of the values

spinnman.messages.scp.scp_result module

class spinnman.messages.scp.scp_result.**SCPResult**(*value*, *doc=''*)

Bases: enum.Enum

The SCP Result codes

Attributes

RC_ARG	Invalid arguments
RC_BUF	No free Shared Memory buffers
RC_CMD	Bad/invalid command
RC_CPU	Bad CPU number
RC_DEAD	SHM destination dead
RC_LEN	Bad packet length
RC_OK	SCPCommand completed OK
RC_P2P_BUSY	Destination busy
RC_P2P_NOREPLY	No reply to open
RC_P2P_REJECT	Open rejected
RC_P2P_TIMEOUT	Dest did not respond
RC_PKT_TX	Pkt Transmission failed
RC_PORT	Bad port number
RC_ROUTE	No P2P route
RC_SUM	Bad checksum
RC_TIMEOUT	Timeout

spinnman.messages.scp.scp_signal module

class spinnman.messages.scp.scp_signal.**SCPSignal**(*value*, *signal_type*, *doc=''*)

Bases: enum.Enum

SCP Signals

Parameters

- **value** (*int*) – The value used for the signal
- **signal_type** (*int*) – The "type" of the signal, between 0 and 2

Attributes

Continued on next page

Table 3.257 – continued from previous page

CONTINUE
EXIT
INITIALISE
PAUSE
POWER_DOWN
START
STOP
SYNC0
SYNC1
TIMER
USER_0
USER_1
USER_2
USER_3

spinnman.messages.sdp package

Submodules

spinnman.messages.sdp.sdp_flag module

class spinnman.messages.sdp.sdp_flag.**SDPFlag**(*value*, *doc=''*)

Bases: enum.Enum

SDPFlag for the message

Attributes

REPLY_EXPECTED	Indicates that a reply is expected
REPLY_NOT_EXPECTED	Indicates that a reply is not expected

spinnman.messages.sdp.sdp_header module

class spinnman.messages.sdp.sdp_header.**SDPHeader**(*flags=None*, *tag=None*, *destination_port=None*, *destination_cpu=None*, *destination_chip_x=None*, *destination_chip_y=None*, *source_port=None*, *source_cpu=None*, *source_chip_x=None*, *source_chip_y=None*)

Bases: object

Represents the header of an SDP message. Each optional parameter in the constructor can be set to a value other than None once, after which it is immutable. It is an error to set a parameter that is not currently None.

Parameters

- **flags** (spinnman.messages.sdp.sdp_flag.SDPFlag) – Any flags for the packet
- **tag** (*int*) – The ip tag of the packet between 0 and 255, or None if it is to be set later
- **destination_port** (*int*) – The destination port of the packet between 0 and 7
- **destination_cpu** (*int*) – The destination processor id within the chip between 0 and 31

- **destination_chip_x** (*int*) – The x-coordinate of the destination chip between 0 and 255
- **destination_chip_y** (*int*) – The y-coordinate of the destination chip between 0 and 255
- **source_port** (*int*) – The source port of the packet between 0 and 7, or None if it is to be set later
- **source_cpu** (*int*) – The source processor id within the chip between 0 and 31, or None if it is to be set later
- **source_chip_x** (*int*) – The x-coordinate of the source chip between 0 and 255, or None if it is to be set later
- **source_chip_y** – The y-coordinate of the source chip between 0 and 255, or None if it is to be set later

Raises spinnman.exceptions.SpinnmanInvalidParameterException If one ofthe parameters is not valid

Attributes

`destination_chip_x`	The x-coordinate of the destination chip of the packet
`destination_chip_y`	The y-coordinate of the destination chip of the packet
`destination_cpu`	The core on the destination chip
`destination_port`	The destination port of the packet
`flags`	The flags of the packet
`source_chip_x`	The x-coordinate of the source chip of the packet
`source_chip_y`	The y-coordinate of the source chip of the packet
`source_cpu`	The core on the source chip
`source_port`	The source port of the packet
`tag`	The tag of the packet

Methods

`read_sdp_header`(byte_reader)	Read an SDP header from a byte_reader
`write_sdp_header`(byte_writer)	Write the SDP header to a byte_writer

Detailed Methods

`read_sdp_header`(*byte_reader*)

Read an SDP header from a byte_reader

Parameters **byte_reader** (`spinnman.data.abstract_byte_reader.AbstractByteReader`) – The reader to read the data from

Returns Nothing is returned

Return type None

Raises

- spinnman.exceptions.SpinnmanIOException – If there is an error reading from the reader
- spinnman.exceptions.SpinnmanInvalidPacketException – If there are too few bytes to read the header

- **spinnman.exceptions.SpinnmanInvalidParameterException** – If there is an error setting any of the values

write_sdp_header(*byte_writer*)
Write the SDP header to a byte_writer

Parameters **byte_writer** (spinnman.data.abstract_byte_writer.AbstractByteWriter) – The writer to write the data to

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error writing to the writer
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If any of the parameter values have not been set

spinnman.messages.sdp.sdp_message module

class spinnman.messages.sdp.sdp_message.**SDPMessage**(*sdp_header*, *data=None*)
Bases: object

Wraps up an SDP message with a header and optional data.

Parameters

- **sdp_header** (spinnman.messages.sdp.sdp_header.SDPHeader) – The header of the message
- **data** (*bytearray*) – The data of the SDP packet, or None if no data

Raises None No known exceptions are thrown

Attributes

data	The data in the packet
sdp_header	The header of the packet

spinnman.messages.spinnaker_boot package

Submodules

spinnman.messages.spinnaker_boot.spinnaker_boot_message module

class spinnman.messages.spinnaker_boot.spinnaker_boot_message.**SpinnakerBootMessage**(*opcode*, *operand_1*, *operand_2*, *operand_3*, *data=None*)
Bases: object

A message used for booting the board

Parameters

- **opcode** (spinnman.messages.spinnaker_boot.spinnaker_boot_op_code.SpinnakerBootOp – The operation of this packet

- **operand_1** (*int*) – The first operand
- **operand_2** (*int*) – The second operand
- **operand_3** (*int*) – The third operand
- **data** (*bytearray*) – The optional data, up to 256 words

Raises spinnman.exceptions.SpinnmanInvalidParameterException If the opcode is not a valid value

Attributes

data	The data
opcode	The operation of this packet
operand_1	The first operand
operand_2	The second operand
operand_3	The third operand

spinnman.messages.spinnaker_boot.spinnaker_boot_messages module

class spinnman.messages.spinnaker_boot.spinnaker_boot_messages.**SpinnakerBootMessages**(*board_vers*

Bases: object

Represents a set of boot messages to be sent to boot the board

Parameters **board_version** (*int*) – The version of the board to be booted

Raises

- spinnman.exceptions.SpinnmanInvalidParameterException – If the board version is not supported
- spinnman.exceptions.SpinnmanIOException – If there is an error assembling the packets

Attributes

messages	Get an iterable of message to be sent.

spinnman.messages.spinnaker_boot.spinnaker_boot_op_code module

class spinnman.messages.spinnaker_boot.spinnaker_boot_op_code.**SpinnakerBootOpCode**(*value*, *doc=''*)

Bases: enum.Enum

Boot message Operation Codes

Attributes

FLOOD_FILL_BLOCK
FLOOD_FILL_CONTROL
FLOOD_FILL_START
HELLO

spinnman.messages.udp_utils package

Submodules

spinnman.messages.udp_utils.udp_utils module

Functions

update_sdp_header(sdp_header, default_sdp_tag)	Apply defaults to the sdp header where the values have not been set

spinnman.messages.udp_utils.udp_utils.**update_sdp_header**(*sdp_header*, *default_sdp_tag*)

Apply defaults to the sdp header where the values have not been set

Parameters **sdp_header** (spinnman.messages.sdp.sdp_header.SDPHeader) – The SDP header values

Returns Nothing is returned

Return type None

Raises spinnman.exceptions.SpinnmanInvalidParameterException If the packet already has a source_port != 7, a source_cpu != 31, a source_chip_x != 0, or a source_chip_y != 0

Submodules

spinnman.messages.multicast_message module

class spinnman.messages.multicast_message.**MulticastMessage**(*key*, *payload=None*)

Bases: object

A SpiNNaker Multicast message

Parameters

- **key** (*int*) – The key of the packet
- **payload** (*int*) – The optional payload of the packet

Raises None No known exceptions are raised

Attributes

key	The key of the packet
payload	The payload of the packet if there is one

spinnman.model package

Submodules

spinnman.model.chip_info module

class spinnman.model.chip_info.**ChipInfo**(*system_data*)

Bases: object

Represents the system variables for a chip, received from the chip SDRAM

Parameters **system_data** (*bytearray*) – An array of bytes retrieved from SDRAM on the baord

Raises spinnman.exceptions.SpinnmanInvalidParameterException If the data doesn't contain valid system data information

Attributes

cpu_clock_mhz	The speed of the CPU clock in MHz
cpu_information_base_address	The base address of the cpu information structure
first_free_router_entry	The id of the first free routing entry on the chip
iobuf_size	The size of the iobuf buffers in bytes
ip_address	The ip address of the chip, or None if no ethernet
is_ethernet_available	True if the ethernet is running on this chip, False otherwise
links_available	The links that are available on the chip
nearest_ethernet_x	The x-coordinate of the nearest chip with ethernet
nearest_ethernet_y	The y-coordinate of the nearest chip with ethernet
physical_to_virtual_core_map	The physical core id to virtual core id map; entries with a value of 0xFF are non-operatio
sdram_base_address	The base address of SDRAM on the chip
virtual_core_ids	A list of available cores by virtual core id (including the monitor)
x	The x-coordinate of the chip
x_size	The number of chips in the x-dimension
y	The y-coordinate of the chip
y_size	The number of chips in the y-dimension

Methods

router_table_copy_address()	The address of the copy of the router table

Detailed Methods

router_table_copy_address()
The address of the copy of the router table

Return type int

spinnman.model.core_subset module

class spinnman.model.core_subset.**CoreSubset**(*x*, *y*, *processor_ids*)
Bases: object

Represents a subset of the cores on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip
- **y** (*int*) – The y-coordinate of the chip
- **processor_ids** (*iterable of int*) – An iterable of processor ids on the chip

Raises spinnman.exceptions.SpinnmanInvalidParameterException If there is more than one core listed with the same id

Attributes

processor_ids	The subset of processor ids on the chip
x	The x-coordinate of the chip
y	The y-coordinate of the chip

Methods

add_processor(processor_id)	Adds a processor id to this subset

Detailed Methods

add_processor (*processor_id*)
Adds a processor id to this subset

Parameters **processor_ids** (*int*) – A processor id

Returns Nothing is returned

Return type None

Raises **spinnman.exceptions.SpinnmanInvalidParameterException** If there is already a processor in the subset with the same id

spinnman.model.core_subsets module

class spinnman.model.core_subsets.**CoreSubsets** (*core_subsets=None*)
Bases: object

Represents a group of CoreSubsets, with a maximum of one per chip

Parameters **core_subsets** (iterable of spinnman.model.core_subset.CoreSubset) – An iterable of cores for each desired chip

Raises **spinnman.exceptions.SpinnmanInvalidParameterException** If there is more than one subset with the same core x and y coordinates

Attributes

core_subsets	The one-per-chip subsets

Methods

add_core_subset(core_subset)	Add a core subset to the set
add_processor(x, y, processor_id)	Add a processor on a given chip to the set
is_chip(x, y)	Determine if the chip with coordinates (x, y) is in the subset
is_core(x, y, processor_id)	Determine if there is a chip with coordinates (x, y) in the subset, which has a core with the g

Detailed Methods

add_core_subset (*core_subset*)
Add a core subset to the set

Parameters **core_subset** (spinnman.model.core_subset.CoreSubset) – The core subset to add

Returns Nothing is returned

Return type None

Raises spinnman.exceptions.SpinnmanInvalidParameterException If there is already a subset with the same core x and y coordinates

add_processor(*x*, *y*, *processor_id*)
Add a processor on a given chip to the set

Parameters

- **x** (*int*) – The x-coordinate of the chip
- **y** (*int*) – The y-coordinate of the chip
- **processor_id** – A processor id

Returns Nothing is returned

Return type None

Raises spinnman.exceptions.SpinnmanInvalidParameterException If there is already a processor on the given chip with the same id

is_chip(*x*, *y*)
Determine if the chip with coordinates (x, y) is in the subset

Parameters

- **x** (*int*) – The x-coordinate of a chip
- **y** (*int*) – The y-coordinate of a chip

Returns True if the chip with coordinates (x, y) is in the subset

Return type bool

is_core(*x*, *y*, *processor_id*)
Determine if there is a chip with coordinates (x, y) in the subset, which has a core with the given id in the subset

Parameters

- **x** (*int*) – The x-coordinate of a chip
- **y** (*int*) – The y-coordinate of a chip
- **processor_id** (*int*) – The id of a core

Returns True if there is a chip with coordinates (x, y) in the subset, which has a core with the given id in the subset

spinnman.model.cpu_info module

class spinnman.model.cpu_info.**CPUInfo**(*x*, *y*, *p*, *cpu_data*)
Bases: object

Represents information about the state of a CPU

Parameters

- **x** (*int*) – The x-coordinate of a chip
- **y** (*int*) – The y-coordinate of a chip

- **p** (*int*) – The id of a core on the chip
- **cpu_data** (*bytearray*) – An array of bytes received from SDRAM on the board

Raises spinnman.exceptions.SpinnmanInvalidParameterException If the array does not contain a cpu data structure

Attributes

application_id	The id of the application running on the core
application_mailbox_command	The command currently in the mailbox being sent from the monitor processor to th
application_mailbox_data_address	The address of the data in SDRAM for the application mailbox
application_name	The name of the application running on the core
iobuf_address	The address of the IOBUF buffer in SDRAM
link_register	The current link register value (lr)
monitor_mailbox_command	The command currently in the mailbox being sent from the application to the moni
monitor_mailbox_data_address	The address of the data in SDRAM of the monitor mailbox
p	The id of the core on the chip
processor_state_register	The value in the processor state register (psr)
registers	The current register values (r0 - r7)
run_time_error	The reason for a run time error
software_error_count	The number of software errors counted
software_source_filename_address	The address of the filename of the software source
software_source_line_number	The line number of the software source
stack_pointer	The current stack pointer value (sp)
state	The current state of the core
time	The time at which the application started
user	The current user values (user0 - user3)
x	The x-coordinate of the chip containing the core
y	The y-coordinate of the chip containing the core

spinnman.model.cpu_state module

class spinnman.model.cpu_state.**CPUState**(*value*, *doc=''*)

Bases: enum.Enum

SARK CPU States

Attributes

C_MAIN
DEAD
FINSHED
IDLE
INITIALISING
PAUSED
POWERED_DOWN
READY
RUNNING
RUN_TIME_EXCEPTION
SYNC0
SYNC1
WATCHDOG

spinnman.model.diagnostic_filter module

class spinnman.model.diagnostic_filter.**DiagnosticFilter**(*enable_interrupt_on_counter_event*, *match_emergency_routing_status_to_incoming_packet*, *destinations*, *sources*, *payload_statuses*, *default_routing_statuses*, *emergency_routing_statuses*, *packet_types*)

Bases: object

A router diagnostic counter filter, which counts packets passing through the router with certain properties. The counter will be incremented so long as the packet matches one of the values in each field i.e. one of each of the destinations, sources, payload_statuses, default_routing_statuses, emergency_routing_statuses and packet_types

Parameters

- **enable_interrupt_on_counter_event** (*bool*) – Indicates whether an interrupt should be raised when this rule matches
- **match_emergency_routing_status_to_incoming_packet** (*bool*) – Indicates whether the emergency routing statuses should be matched against packets arriving at this router (if True), or if they should be matched against packets leaving this router (if False)
- **destinations** (iterable of spinnman.model.diagnostic_filter_destination.DiagnosticFilte – Increment the counter if one or more of the given destinations match
- **sources** (iterable of spinnman.model.diagnostic_filter_source.DiagnosticFilterSource) – Increment the counter if one or more of the given sources match (or None or empty list to match all)
- **payload_statuses** (iterable of spinnman.model.diagnostic_filter_payload_status.Diagnosti – Increment the counter if one or more of the given payload statuses match (or None or empty list to match all)
- **default_routing_statuses** (iterable of spinnman.model.diagnostic_filter_default_routing_st – Increment the counter if one or more of the given default routing statuses match (or None or empty list to match all)
- **emergency_routing_statuses** (iterable of spinnman.model.diagnostic_filter_emergency_routin – Increment the counter if one or more of the given emergency routing statuses match (or None or empty list to match all)
- **packet_types** (iterable of spinnman.model.diagnostic_filter_packet_type.DiagnosticFilte – Increment the counter if one or more of the given packet types match (or None or empty list to match all)

Attributes

default_routing_statuses	
destinations	
emergency_routing_statuses	
enable_interrupt_on_counter_event	
filter_word	A word of data that can be written to the router to set up t
match_emergency_routing_status_to_incoming_packet	
packet_types	
payload_statuses	

Continued on nex

Table 3.275 – continued from previous page

sources

Methods

read_from_int(int_value)

Detailed Methods

static read_from_int (*int_value*)

spinnman.model.diagnostic_filter_default_routing_status module

class spinnman.model.diagnostic_filter_default_routing_status.**DiagnosticFilterDefaultRoutingS**

Bases: enum.Enum

Default routing flags for the diagnostic filters. Note that only one has to match for the counter to be incremented

Attributes

DEFAULT_ROUTED	Packet is to be default routed
NON_DEFAULT_ROUTED	Packet is not to be default routed

spinnman.model.diagnostic_filter_destination module

class spinnman.model.diagnostic_filter_destination.**DiagnosticFilterDestination** (*value*, *doc=''*)

Bases: enum.Enum

Destination flags for the diagnostic filters. Note that only one has to match for the counter to be incremented

Attributes

DUMP	Destination is to dump the packet
LINK_0	Destination is link 0
LINK_1	Destination is link 1
LINK_2	Destination is link 2
LINK_3	Destination is link 3
LINK_4	Destination is link 4
LINK_5	Destination is link 5
LOCAL	Destination is a local core (but not the monitor core)
LOCAL_MONITOR	Destination is the local monitor core

spinnman.model.diagnostic_filter_emergency_routing_status module

class spinnman.model.diagnostic_filter_emergency_routing_status.**DiagnosticFilterEmergencyRout**

Bases: enum.Enum

Emergency routing status flags for the diagnostic filters. Note that only one has to match for the counter to be incremented

Attributes

FIRST_STAGE	Packet is in first hop of emergency route; packet wouldn't have reached this router without emergency
FIRST_STAGE_COMBINED	Packet is in first hop of emergency route; packet should also have been sent here by normal routing
NORMAL	Packet is not emergency routed
SECOND_STAGE	Packet is in last hop of emergency route and should now return to normal routing

spinnman.model.diagnostic_filter_packet_type module

class spinnman.model.diagnostic_filter_packet_type.**DiagnosticFilterPacketType**(*value*, *doc=''*)

Bases: enum.Enum

Packet type flags for the diagnostic filters. Note that only one has to match for the counter to be incremented

Attributes

FIXED_ROUTE	Packet is fixed-route
MULTICAST	Packet is multicast
NEAREST_NEIGHBOUR	Packet is nearest-neighbour
POINT_TO_POINT	Packet is point-to-point

spinnman.model.diagnostic_filter_payload_status module

class spinnman.model.diagnostic_filter_payload_status.**DiagnosticFilterPayloadStatus**(*value*, *doc=''*)

Bases: enum.Enum

Payload flags for the diagnostic filters. Note that only one has to match for the counter to be incremented

Attributes

WITHOUT_PAYLOAD	Packet doesn't have a payload
WITH_PAYLOAD	Packet has a payload

spinnman.model.diagnostic_filter_source module

class spinnman.model.diagnostic_filter_source.**DiagnosticFilterSource**(*value*, *doc=''*)

Bases: enum.Enum

Source flags for the diagnostic filters. Note that only one has to match for the counter to be incremented

Attributes

LOCAL	Source is a local core
NON_LOCAL	Source is not a local core

spinnman.model.io_buffer module

class spinnman.model.io_buffer.**IOBuffer**(*x*, *y*, *p*, *iobuf*)

Bases: object

The contents of IOBUF for a core

Parameters

- **x** (*int*) – The x-coordinate of a chip
- **y** (*int*) – The y-coordinate of a chip
- **p** (*int*) – The p-coordinate of a chip
- **iobuf** (*str*) – The contents of the buffer for the chip

Raises None No known exceptions are raised

Attributes

`iobuf`	The contents of the buffer
`p`	The id of the core on the chip
`x`	The x-coordinate of the chip containing the core
`y`	The y-coordinate of the chip containing the core

spinnman.model.machine_dimensions module

class `spinnman.model.machine_dimensions.`**`MachineDimensions`**(*x_max*, *y_max*)

Bases: `object`

Represents the dimensions of a machine

Parameters

- **x_max** (*int*) – The maximum x-coordinate of the chips in the machine
- **y_max** (*int*) – The maximum y-coordinate of the chips in the machine

Raises None No known exceptions are raised

Attributes

`x_max`	The maximum x-coordinate of the chips in the machine
`y_max`	The maximum y-coordinate of the chips in the machine

spinnman.model.mailbox_command module

class `spinnman.model.mailbox_command.`**`MailboxCommand`**(*value*, *doc='‘*)

Bases: `enum.Enum`

Commands sent between an application and the monitor processor

Attributes

`SHM_CMD`	The mailbox contains a command
`SHM_IDLE`	The mailbox is idle
`SHM_MSG`	The mailbox contains an SDP message
`SHM_NOP`	The mailbox contains a non-operation
`SHM_SIGNAL`	The mailbox contains a signal

spinnman.model.router_diagnostics module

class spinnman.model.router_diagnostics.**RouterDiagnostics**(*control_register*, *error_status*, *register_values*)

Bases: object

Represents a set of diagnostic information available from a chip router

Parameters

- **control_register** (*int*) – The value of the control register
- **error_status** (*int*) – The value of the error_status
- **register_values** (*iterable of int*) – The values of the 16 router registers

Raises spinnman.exceptions.SpinnmanInvalidParameterException If the number of register values is not 16

Attributes

error_status	The error status
mon	The "mon" part of the control register
n_dropped_fixed_route_packets	The number of fixed-route packets received that were dropped
n_dropped_multicast_packets	The number of multicast packets received that were dropped
n_dropped_nearest_neighbour_packets	The number of nearest-neighbour packets received that were dropped
n_dropped_peer_to_peer_packets	The number of peer-to-peer packets received that were dropped
n_external_fixed_route_packets	The number of fixed-route packets received from external links
n_external_multicast_packets	The number of multicast packets received from external links
n_external_nearest_neighbour_packets	The number of nearest-neighbour packets received from external links
n_external_peer_to_peer_packets	The number of peer-to-peer packets received from external links
n_local_fixed_route_packets	The number of fixed-route packets received from local cores
n_local_multicast_packets	The number of multicast packets received from local cores
n_local_nearest_neighbour_packets	The number of nearest-neighbour packets received from local cores
n_local_peer_to_peer_packets	The number of peer-to-peer packets received from local cores
registers	The values in all of the registers.
user_registers	The values in the user control registers
wait_1	The wait_1 part of the control register
wait_2	The wait_2 part of the control register

spinnman.model.run_time_error module

class spinnman.model.run_time_error.**RunTimeError**(*value*, *doc='*')

Bases: enum.Enum

SARK Run time errors

Attributes

ABORT
API
DABT
DIVBY0
ENABLE
EVENT

Continued on next page

Table 3.287 – continued from previous page

FIQ
IOBUF
IRQ
MALLOC
NONE
NULL
PABT
PKT
RESET
SVC
SWERR
TIMER
UNDEF
VIC

spinnman.model.version_info module

Functions

asctime(([tuple]) -> string)	Convert a time tuple to a string, e.g.
localtime(...)	tm_sec,tm_wday,tm_yday,tm_isdst)

spinnman.model.version_info.**asctime**([*tuple*]) → string
Convert a time tuple to a string, e.g. 'Sat Jun 06 16:26:11 1998'. When the time tuple is not present, current time as returned by localtime() is used.

spinnman.model.version_info.**localtime**([*seconds*]) → (tm_year,tm_mon,tm_mday,tm_hour,tm_min, tm_sec,tm_wday,tm_yday,tm_isdst)

Convert seconds since the Epoch to a time tuple expressing local time. When 'seconds' is not passed in, convert the current time instead.

class spinnman.model.version_info.**VersionInfo**(*version_data*)
Bases: object

Decodes SC&MP/SARK version information as returned by the SVER command

Parameters **version_data** (*bytearray*) – bytes from an SCP packet containing version information

Raises **spinnman.exceptions.SpinnmanInvalidParameterException** If the message does not contain valid version information

Attributes

build_date	The build date of the software
hardware	The hardware being run on
name	The name of the software
p	The processor id of the processor where the information was obtained
version_number	The version number of the software
version_string	The version information as text
x	The x-coordinate of the chip where the information was obtained

Continued on next page

Table 3.289 – continued from previous page

y	The y-coordinate of the chip where the information was obtained

3.1.2 Submodules

spinnman.constants module

Classes

CONNECTION_TYPE
EIEIO_COMMAND_IDS
TRAFFIC_TYPE

class spinnman.constants.**CONNECTION_TYPE**
Bases: enum.Enum

Attributes

REVERSE_IPTAG
SDP_IPTAG
UDP_BOOT
UDP_IPTAG
UDP_SPINNAKER
USB

class spinnman.constants.**EIEIO_COMMAND_IDS**
Bases: enum.Enum

Attributes

DATABASE_CONFIRMATION
EVENT_PADDING
EVENT_STOP
HOST_DATA_READ
HOST_SEND_SEQUENCED_DATA
SPINNAKER_REQUEST_BUFFERS
SPINNAKER_REQUEST_READ_DATA
START_SENDING_REQUESTS
STOP_SENDING_REQUESTS

class spinnman.constants.**TRAFFIC_TYPE**
Bases: enum.Enum

Attributes

EIEIO_COMMAND
EIEIO_DATA
SCP
SDP
UDP

spinnman.exceptions module

Exceptions

SpinnmanEIEIOPacketParsingException(...)	Unable to complete the parsing of the EIEIO packet received.
SpinnmanException	Superclass of exceptions that occur when dealing with communi
SpinnmanIOException(problem)	An exception that something went wrong with the underlying IO
SpinnmanInvalidPacketException(packet_type, ...)	An exception that indicates that a packet was not in the expected
SpinnmanInvalidParameterException(parameter, ...)	An exception that indicates that the value of one of the paramete
SpinnmanInvalidParameterTypeException(...)	An exception that indicates that the type of one of the parameter
SpinnmanTimeoutException(operation, timeout)	An exception that indicates that a timeout occurred before an op
SpinnmanUnexpectedResponseCodeException(...)	Indicate that a response code returned from the board was unexp
SpinnmanUnsupportedOperationException(operation)	An exception that indicates that the given operation is not suppo

exception spinnman.exceptions.**SpinnmanEIEIOPacketParsingException**(*parsing_format*, *packet*)

Unable to complete the parsing of the EIEIO packet received. The routine used is invalid or the content of the packet is invalid

exception spinnman.exceptions.**SpinnmanException**

Superclass of exceptions that occur when dealing with communication with SpiNNaker

exception spinnman.exceptions.**SpinnmanIOException**(*problem*)

An exception that something went wrong with the underlying IO

Parameters **problem** (*str*) – The problem with the IO

exception spinnman.exceptions.**SpinnmanInvalidPacketException**(*packet_type*, *problem*)

An exception that indicates that a packet was not in the expected format

Parameters

- **packet_type** (*str*) – The type of packet expected
- **problem** (*str*) – The problem with the packet

exception spinnman.exceptions.**SpinnmanInvalidParameterException**(*parameter*, *value*, *problem*)

An exception that indicates that the value of one of the parameters passed was invalid

Parameters

- **parameter** (*str*) – The name of the parameter that is invalid
- **value** (*str*) – The value of the parameter that is invalid
- **problem** (*str*) – The problem with the parameter

exception spinnman.exceptions.**SpinnmanInvalidParameterTypeException**(*parameter*, *param_type*, *problem*)

An exception that indicates that the type of one of the parameters passed was invalid

Parameters

- **parameter** (*str*) – The name of the parameter that is invalid
- **param_type** (*str*) – The type of the parameter that is invalid
- **problem** (*str*) – The problem with the parameter

exception spinnman.exceptions.**SpinnmanTimeoutException**(*operation*, *timeout*)
An exception that indicates that a timeout occurred before an operation could finish

Parameters

- **operation** (*str*) – The operation being performed
- **timeout** (*int*) – The timeout value in seconds

exception spinnman.exceptions.**SpinnmanUnexpectedResponseCodeException**(*operation*, *command*, *response*)
Indicate that a response code returned from the board was unexpected for the current operation

Parameters

- **operation** (*str*) – The operation being performed
- **command** (*str*) – The command being executed
- **response** (*str*) – The response received in error

exception spinnman.exceptions.**SpinnmanUnsupportedOperationException**(*operation*)
An exception that indicates that the given operation is not supported

Parameters **operation** (*str*) – The operation being requested

spinnman.reports module

Functions

generate_machine_report(report_directory, ...)	Generate report on the physical structure of the target SpiNNaker machine.

spinnman.reports.**generate_machine_report**(*report_directory*, *machine*, *connections*)
Generate report on the physical structure of the target SpiNNaker machine.

Parameters

- **report_directory** (*str*) – the directroy to which reports are stored
- **machine** (*spinnmachine.machine.Machine object*) – the machine python object
- **connections** (*iterable of implientations of*) – the list of connections to the machine

spinnman.connections.abstract_connection.AbstractConnection :return None :rtype: None :raise IOError: when a file cannot be opened for some reason

spinnman.transceiver module

Functions

create_transceiver_from_hostname(hostname[, ...])	Create a Transceiver by creating a UDPConnection to the given hos
gethostbyname((host) -> address)	Return the IP address (a string of the form '255.255.255.255') for a
inet_aton(...)	Convert an IP address in string format (123.45.67.89) to the 32-bit

spinnman.transceiver.**create_transceiver_from_hostname**(*hostname*, *ignore_chips=None*, *ignore_cores=None*, *max_core_id=None*)

Create a Transceiver by creating a UDPConnection to the given hostname on port 17893 (the default SCAMP port), and a
optionally discovering any additional links using the UDPConnection, and then returning the transceiver created with the conjunction of the created UDPConnection and the discovered connections

Parameters

- **hostname** (*str*) – The hostname or IP address of the board
- **ignore_chips** (spinnman.model.core_subsets.CoreSubsets) – An optional set of chips to ignore in the machine. Requests for a "machine" will have these chips excluded, as if they never existed. The processor_ids of the specified chips are ignored.
- **ignore_cores** (spinnman.model.core_subsets.CoreSubsets) – An optional set of cores to ignore in the machine. Requests for a "machine" will have these cores excluded, as if they never existed.
- **max_core_id** (*int*) – The maximum core id in any discovered machine. Requests for a "machine" will only have core ids up to this value.

Returns The created transceiver

Return type spinnman.transceiver.Transceiver

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

spinnman.transceiver.**gethostbyname**(*host*) → address
Return the IP address (a string of the form '255.255.255.255') for a host.

spinnman.transceiver.**inet_aton**(*string*) → packed 32-bit IP representation
Convert an IP address in string format (123.45.67.89) to the 32-bit packed binary format used in low-level network functions.

class spinnman.transceiver.**Transceiver**(*connections=None*, *ignore_chips=None*, *ignore_cores=None*, *max_core_id=None*, *shut_down_connections=False*, *n_scp_threads=16*, *n_other_threads=16*)

Bases: object

An encapsulation of various communications with the spinnaker board.

The methods of this class are designed to be thread-safe; thus you can make multiple calls to the same (or different) methods from multiple threads and expect each call to work as if it had been called sequentially, although the order of returns is not guaranteed. Note also that with multiple connections to the board, using multiple threads in this way may result in an increase in the overall speed of operation, since the multiple calls may be made separately over the set of given connections.

Parameters

- **connections** (iterable of `spinnman.connections.abstract_connection.AbstractConnection`) – An iterable of connections to the board. If not specified, no communication will be possible until connections are found.
- **ignore_chips** (`spinnman.model.core_subsets.CoreSubsets`) – An optional set of chips to ignore in the machine. Requests for a "machine" will have these chips excluded, as if they never existed. The processor_ids of the specified chips are ignored.
- **ignore_cores** (`spinnman.model.core_subsets.CoreSubsets`) – An optional set of cores to ignore in the machine. Requests for a "machine" will have these cores excluded, as if they never existed.
- **max_core_id** (*int*) – The maximum core id in any discovered machine. Requests for a "machine" will only have core ids up to and including this value.

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board, or if no connections to the board can be found (if connections is None)
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

Methods

`boot_board`(board_version)	Attempt to boot the board.
`clear_ip_tag`(tag[, connection, board_address])	Clear the setting of an ip tag
`clear_multicast_routes`(x, y)	Remove all the multicast routes on a chip
`clear_router_diagnostic_counters`(x, y[, ...])	Clear router diagnostic information om a chip
`close`([close_original_connections])	Close the transceiver and any threads that are running
`discover_scamp_connections`()	Find connections to the board and store these for future use.
`ensure_board_is_ready`(board_version[, n_retries])	Ensure that the board is ready to interact with this version of
`execute`(x, y, processors, executable, app_id)	Start an executable running on a single core
`execute_flood`(core_subsets, executable, app_id)	Start an executable running on multiple places on the board.
`get_connections`([include_boot_connection])	Get the currently known connections to the board, made up
`get_core_state_count`(app_id, state)	Get a count of the number of cores which have a given state
`get_cpu_information`([core_subsets])	Get information about the processors on the board
`get_cpu_information_from_core`(x, y, p)	Get information about a specific processor on the board
`get_iobuf`([core_subsets])	Get the contents of the IOBUF buffer for a number of proces
`get_iobuf_from_core`(x, y, p)	Get the contents of IOBUF for a given core
`get_machine_details`()	Get the details of the machine made up of chips on a board a
`get_machine_dimensions`()	Get the maximum chip x-coordinate and maximum chip y-c
`get_multicast_routes`(x, y[, app_id])	Get the current multicast routes set up on a chip

Table 3.297 – continued from previou

get_router_diagnostic_filter(x, y, position)	Gets a router diagnostic filter from a router
get_router_diagnostics(x, y)	Get router diagnostic information from a chip
get_scamp_version([n_retries, timeout])	Get the version of scamp which is running on the board
get_tags([connection])	Get the current set of tags that have been set on the board
get_user_0_register_address_from_core(x, y, p)	Get the address of user 0 for a given processor on the board
is_connected([connection])	Determines if the board can be contacted
load_multicast_routes(x, y, routes, app_id)	Load a set of multicast routes on to a chip
locate_spinnaker_connection_for_board_address(...)	Find a connection that matches the given board IP address
read_memory(x, y, base_address, length)	Read some areas of SDRAM from the board
read_memory_return_byte_array(x, y, ...)	Read some areas of SDRAM from the board
read_neighbour_memory(x, y, cpu, link, ...)	Read some areas of memory on a neighbouring chip using a
receive_multicast_message(x, y[, timeout, ...])	Receives a multicast message from the board
register_listener(callback, recieve_port_no, ...)	Register a callback for a certain type of traffic
send_eieio_command_message(message[, connection])	Sends a EIEIO command message using one of the connecti
send_multicast_message(x, y, multicast_message)	Sends a multicast message to the board
send_scp_message(message[, retry_codes, ...])	Sends an SCP message, and gets a response
send_sdp_message(message[, connection])	Sends a EIEIO command message using one of the connecti
send_signal(app_id, signal)	Send a signal to an application
set_ip_tag(ip_tag)	Set up an ip tag
set_leds(x, y, cpu, led_states)	Set LED states.
set_reverse_ip_tag(reverse_ip_tag)	Set up a reverse ip tag
set_router_diagnostic_filter(x, y, position, ...)	Sets a router diagnostic filter in a router
stop_application(app_id)	Sends a stop request for an app_id
write_memory(x, y, base_address, data[, n_bytes])	Write to the SDRAM on the board
write_memory_flood(base_address, data[, n_bytes])	Write to the SDRAM of all chips.
write_neighbour_memory(x, y, cpu, link, ...)	Write to the memory of a neighbouring chip using a LINK_

Detailed Methods

boot_board(*board_version*)
Attempt to boot the board. No check is performed to see if the board is already booted.

Parameters **board_version** (*int*) – The version of the board e.g. 3 for a SpiNN-3 board or 5 for a SpiNN-5 board.

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanInvalidParameterException** – If the board version is not known
- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board

clear_ip_tag(*tag*, *connection=None*, *board_address=None*)
Clear the setting of an ip tag

Parameters

- **tag** (*int*) – The tag id
- **connection** (spinnman.connections.abstract_scp_sender.AbstractSCPSender) – Connection where the tag should be cleard. If not specified, all SCPSender connections will send the message to clear the tag

- **board_address** – Board address where the tag should be cleared. If not specified, all SCPSender connections will send the message to clear the tag

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If the tag is not a valid tag
  - If the connection cannot send SDP messages
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

clear_multicast_routes(*x*, *y*)

Remove all the multicast routes on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip on which to clear the routes
- **y** (*int*) – The y-coordinate of the chip on which to clear the routes

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

clear_router_diagnostic_counters(*x, y, enable=True, counter_ids=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]*)

Clear router diagnostic information om a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip
- **y** (*int*) – The y-coordinate of the chip
- **enable** (*bool*) – True (default) if the counters should be enabled
- **counter_ids** (*array-like of int*) – The ids of the counters to reset (all by default) and enable if enable is True; each must be between 0 and 15

Returns None

Return type Nothing is returned

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters or a counter id is out of range
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

close(*close_original_connections=True*)

Close the transceiver and any threads that are running

Parameters **close_original_connections** – If True, the original connections passed to the transceiver in the constructor are also closed. If False, only newly discovered connections are closed.

Returns Nothing is returned

Return type None

Raises None No known exceptions are raised

discover_scamp_connections()

Find connections to the board and store these for future use. Note that connections can be empty, in which case another local discovery mechanism will be used. Note that an exception will be thrown if no initial connections can be found to the board.

Returns An iterable of discovered connections, not including the initially given connections in the constructor

Return type iterable of `spinnman.connections.abstract_connection.AbstractConnection`

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

ensure_board_is_ready(*board_version*, *n_retries=3*)

Ensure that the board is ready to interact with this version of the transceiver. Boots the board if not already booted and verifies that the version of SCAMP running is compatible with this transceiver.

Parameters

- **board_version** (*int*) – The version of the board e.g. 3 for a SpiNN-3 board or 5 for a SpiNN-5 board.
- **n_retries** (*int*) – The number of times to retry booting

Returns The version identifier

Return type spinnman.model.version_info.VersionInfo

Raise spinnman.exceptions.SpinnmanIOException: * If there is a problem booting the board * If the version of software on the board is not compatible with this transceiver

execute(*x*, *y*, *processors*, *executable*, *app_id*, *n_bytes=None*)

Start an executable running on a single core

Parameters

- **x** (*int*) – The x-coordinate of the chip on which to run the executable
- **y** (*int*) – The y-coordinate of the chip on which to run the executable
- **processors** (*iterable of int*) – The cores on the chip on which to run the application
- **executable** (spinnman.data.abstract_data_reader.AbstractDataReader or bytearray) – The data that is to be executed. Should be one of the following: * An instance of AbstractDataReader * A bytearray
- **app_id** (*int*) – The id of the application with which to associate the executable
- **n_bytes** (*int*) – The size of the executable data in bytes. If not specified: * If data is an AbstractDataReader, an error is raised * If data is a bytearray, the length of the bytearray will be used * If data is an int, 4 will be used

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** –
  - If there is an error communicating with the board
  - If there is an error reading the executable
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If x, y, p does not lead to a valid core
  - If app_id is an invalid application id
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

execute_flood(*core_subsets*, *executable*, *app_id*, *n_bytes=None*)

Start an executable running on multiple places on the board. This will be optimized based on the selected cores, but it may still require a number of communications with the board to execute.

Parameters

- **core_subsets** (spinnman.model.core_subsets.CoreSubsets) – Which cores on which chips to start the executable
- **executable** (spinnman.data.abstract_data_reader.AbstractDataReader or bytearray) – The data that is to be executed. Should be one of the following: * An instance of AbstractDataReader * A bytearray

- **app_id** (*int*) – The id of the application with which to associate the executable
- **n_bytes** (*int*) – The size of the executable data in bytes. If not specified: * If data is an AbstractDataReader, an error is raised * If data is a bytearray, the length of the bytearray will be used * If data is an int, 4 will be used

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** –
  - If there is an error communicating with the board
  - If there is an error reading the executable
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If one of the specified cores is not valid
  - If app_id is an invalid application id
  - If a packet is received that has invalid parameters
  - If data is an AbstractDataReader but n_bytes is not specified
  - If data is an int and n_bytes is more than 4
  - If n_bytes is less than 0
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_connections(*include_boot_connection=False*)

Get the currently known connections to the board, made up of those passed in to the transceiver and those that are discovered during calls to discover_connections. No further discovery is done here.

Parameters **include_boot_connection** (*bool*) – this parameter signals if the returned list of connections should include also the boot connection to SpiNNaker

Returns An iterable of connections known to the transceiver

Return type iterable of `spinnman.connections.abstract_connection.AbstractConnection`

Raises None No known exceptions are raised

get_core_state_count(*app_id*, *state*)

Get a count of the number of cores which have a given state

Parameters

- **app_id** (*int*) – The id of the application from which to get the count.
- **state** (`spinnman.model.cpu_state.CPUState`) – The state count to get

Returns A count of the cores with the given status

Return type int

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board

- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If state is not a valid status
  - If app_id is not a valid application id
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_cpu_information(*core_subsets=None*)

Get information about the processors on the board

Parameters **core_subsets** (spinnman.model.core_subsets.CoreSubsets) – A set of chips and cores from which to get the information. If not specified, the information from all of the cores on all of the chips on the board are obtained

Returns An iterable of the cpu information for the selected cores, or all cores if core_subsets is not specified

Return type iterable of spinnman.model.cpu_info.CPUInfo

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If chip_and_cores contains invalid items
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_cpu_information_from_core(*x*, *y*, *p*)

Get information about a specific processor on the board

Parameters

- **x** (*int*) – The x-coordinate of the chip containing the processor
- **y** (*int*) – The y-coordinate of the chip containing the processor
- **p** (*int*) – The id of the processor to get the information about

Returns The cpu information for the selected core

Return type spinnman.model.cpu_info.CPUInfo

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –

- If x, y, p is not a valid processor
- If a packet is received that has invalid parameters

- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_iobuf(*core_subsets=None*)

Get the contents of the IOBUF buffer for a number of processors

Parameters **core_subsets** (spinnman.model.core_subsets.CoreSubsets) – A set of chips and cores from which to get the buffers. If not specified, the buffers from all of the cores on all of the chips on the board are obtained

Returns An iterable of the buffers, which may not be in the order of core_subsets

Return type iterable of spinnman.model.io_buffer.IOBuffer

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If chip_and_cores contains invalid items
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_iobuf_from_core(*x*, *y*, *p*)

Get the contents of IOBUF for a given core

Parameters

- **x** (*int*) – The x-coordinate of the chip containing the processor
- **y** (*int*) – The y-coordinate of the chip containing the processor
- **p** (*int*) – The id of the processor to get the IOBUF for

Returns An IOBUF buffer

Return type spinnman.model.io_buffer.IOBuffer

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If chip_and_cores contains invalid items
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_machine_details()
Get the details of the machine made up of chips on a board and how they are connected to each other.

Returns A machine description

Return type spinn_machine.machine.Machine

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_machine_dimensions()
Get the maximum chip x-coordinate and maximum chip y-coordinate of the chips in the machine

Returns The dimensions of the machine

Return type spinnman.model.machine_dimensions.MachineDimensions

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_multicast_routes(*x*, *y*, *app_id=None*)
Get the current multicast routes set up on a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip from which to get the routes
- **y** (*int*) – The y-coordinate of the chip from which to get the routes
- **app_id** (*int*) – The id of the application to filter the routes for. If not specified, will return all routes

Returns An iterable of multicast routes

Return type iterable of spinnman.model.multicast_routing_entry.MulticastRoute

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format

- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_router_diagnostic_filter (*x*, *y*, *position*)
Gets a router diagnostic filter from a router

Parameters

- **x** (*int*) – the x address of the router from which this filter is being retrieved
- **y** (*int*) – the y address of the router from which this filter is being retrieved
- **position** (*int*) – the position in the list of filters where this filter is to be added

Returns The diagnostic filter read

Return type `spinnman.model.diagnostic_filter.DiagnosticFilter`

Raises

- **spinnman.exceptions.SpinnmanIOException** –
  - If there is an error communicating with the board
  - If there is an error reading the data
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If x, y does not lead to a valid chip
  - If a packet is received that has invalid parameters
  - If position is less than 0 or more than 15
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_router_diagnostics (*x*, *y*)
Get router diagnostic information from a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip from which to get the information
- **y** (*int*) – The y-coordinate of the chip from which to get the information

Returns The router diagnostic information

Return type `spinnman.model.router_diagnostics.RouterDiagnostics`

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters

- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_scamp_version(*n_retries=3*, *timeout=1*)
Get the version of scamp which is running on the board

Parameters

- **n_retries** (*int*) – The number of times to retry getting the version
- **timeout** (*int*) – The timeout for each retry in seconds

Returns The version identifier

Return type spinnman.model.version_info.VersionInfo

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If the timeout is less than 1
- **spinnman.exceptions.SpinnmanTimeoutException** – If none of the retries resulted in a response before the timeout (suggesting that the board is not booted)

get_tags(*connection=None*)
Get the current set of tags that have been set on the board

Parameters **connection** (spinnman.connections.abstract_scp_sender.AbstractSCPSender) – Connection from which the tags should be received. If not specified, all SCPSender connections will be queried and the response will be combined.

Returns An iterable of tags

Return type iterable of spinn_machine.tags.abstract_tag.AbstractTag

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If the connection cannot send SDP messages
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

get_user_0_register_address_from_core(*x*, *y*, *p*)
Get the address of user 0 for a given processor on the board

Parameters

- **x** (*int*) – the x-coordinate of the chip containing the processor
- **y** (*int*) – the y-coordinate of the chip containing the processor
- **p** (*int*) – The id of the processor to get the user 0 address from

:return:The address for user 0 register for this processor :rtype: int :raise spinnman.exceptions.SpinnmanInvalidPacketException: If a packet is received that is not in the valid format :raise spinnman.exceptions.SpinnmanInvalidParameterException:

- If x, y, p is not a valid processor
- If a packet is received that has invalid parameters

Raises spinnman.exceptions.SpinnmanUnexpectedResponseCodeException If a response indicates an error during the exchange

is_connected(*connection=None*)

Determines if the board can be contacted

Parameters **connection** (spinnman.connections.abstract_connection.AbstractConnection) – The connection which is to be tested. If none, all connections will be tested, and the board will be considered to be connected if any one connection works.

Returns True if the board can be contacted, False otherwise

Return type bool

Raises None No known exceptions are raised

load_multicast_routes(*x*, *y*, *routes*, *app_id*)

Load a set of multicast routes on to a chip

Parameters

- **x** (*int*) – The x-coordinate of the chip onto which to load the routes
- **y** (*int*) – The y-coordinate of the chip onto which to load the routes
- **routes** (iterable of spinnmachine.multicast_routing_entry.MulticastRoutingEntry) – An iterable of multicast routes to load
- **app_id** (*int*) – The id of the application with which to associate the routes. If not specified, defaults to 0.

Returns Nothing is returned

Return type None

Raises

- spinnman.exceptions.SpinnmanIOException – If there is an error communicating with the board
- spinnman.exceptions.SpinnmanInvalidPacketException – If a packet is received that is not in the valid format
- spinnman.exceptions.SpinnmanInvalidParameterException –
  - If any of the routes are invalid
  - If a packet is received that has invalid parameters
- spinnman.exceptions.SpinnmanUnexpectedResponseCodeException – If a response indicates an error during the exchange

locate_spinnaker_connection_for_board_address(*board_address*)

Find a connection that matches the given board IP address

Parameters **board_address** (*str*) – The IP address of the ethernet connection on the baord

Returns A connection for the given IP address, or None if no such connection exists

Return type spinnman.connections.udp_packet_connections.udp_spinnaker_connection.U

read_memory(*x*, *y*, *base_address*, *length*)
Read some areas of SDRAM from the board

Parameters

- **x** (*int*) – The x-coordinate of the chip where the memory is to be read from
- **y** (*int*) – The y-coordinate of the chip where the memory is to be read from
- **base_address** (*int*) – The address in SDRAM where the region of memory to be read starts
- **length** (*int*) – The length of the data to be read in bytes

Returns An iterable of chunks of data read in order

Return type iterable of bytearray

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If one of x, y, p, base_address or length is invalid
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

read_memory_return_byte_array(*x*, *y*, *base_address*, *length*)
Read some areas of SDRAM from the board

Parameters

- **x** (*int*) – The x-coordinate of the chip where the memory is to be read from
- **y** (*int*) – The y-coordinate of the chip where the memory is to be read from
- **base_address** (*int*) – The address in SDRAM where the region of memory to be read starts
- **length** (*int*) – The length of the data to be read in bytes

Returns An full bytearray of data read in order

Return type bytearray

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If one of x, y, p, base_address or length is invalid
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

read_neighbour_memory(*x*, *y*, *cpu*, *link*, *base_address*, *length*)
Read some areas of memory on a neighbouring chip using a LINK_READ SCP command. If sent to a BMP, this command can be used to communicate with the FPGAs' debug registers.

Parameters

- **x** (*int*) – The x-coordinate of the chip whose neighbour is to be read from
- **y** (*int*) – The y-coordinate of the chip whose neighbour is to be read from
- **cpu** (*int*) – The cpu to use, typically 0 (or if a BMP, the slot number)
- **link** – The link index to send the request to (or if BMP, the FPGA

number) :type link: int :param base_address: The address in SDRAM where the region of memory to be read starts :type base_address: int :param length: The length of the data to be read in bytes :type length: int :return: An iterable of chunks of data read in order :rtype: iterable of bytearray :raise spinnman.exceptions.SpinnmanIOException: If there is an error communicating with the board :raise spinnman.exceptions.SpinnmanInvalidPacketException: If a packet is received that is not in the valid format :raise spinnman.exceptions.SpinnmanInvalidParameterException:

•If one of x, y, p, base_address or length is invalid

•If a packet is received that has invalid parameters

Raises **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** If a response indicates an error during the exchange

receive_multicast_message(*x*, *y*, *timeout=None*, *connection=None*)
Receives a multicast message from the board

Parameters

- **x** (*int*) – The x-coordinate of the chip where the message should come from on the board
- **y** (*int*) – The y-coordinate of the chip where the message should come from on the board
- **timeout** (*int*) – Amount of time to wait for the message to arrive in seconds before a timeout. If not specified, will wait indefinitely, or until the selected connection is closed
- **connection** (`spinnman.connections.abstract_multicast_receiver.AbstractMulticastI` – A specific connection from which to receive the message. If not specified, an appropriate connection is chosen automatically

Returns The received message

Return type `spinnman.messages.multicast_message.MulticastMessage`

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanUnsupportedOperationException** –
  - If there is no connection that supports reception over multicast (or the given connection does not)
  - If there is no connection that can receive a packet from the selected chip (ignoring routing tables)
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If the message received is not a valid multicast message
- **spinnman.exceptions.SpinnmanInvalidParameterException** –

- If the timeout value is not valid
- If the received packet has an invalid parameter

register_listener(*callback*, *recieve_port_no*, *connection_type*, *traffic_type*, *hostname=None*)
Register a callback for a certain type of traffic

Parameters

- **callback** (*function(packet)*) – Function to be called when a packet is received
- **recieve_port_no** (*int*) – The port number to listen on
- **connection_type** – The type of the connection
- **traffic_type** – The type of traffic expected on the connection
- **hostname** (*str*) – The optional hostname to listen on

send_eieio_command_message(*message*, *connection=None*)
Sends a EIEIO command message using one of the connections.

Parameters

- **message** (*EIEIOCommandMessage*) – The message to send
- **connection** (spinnman.connections.abstract_connection.AbstractConnection) – An optional connection to use

Returns None

send_multicast_message(*x*, *y*, *multicast_message*, *connection=None*)
Sends a multicast message to the board

Parameters

- **x** (*int*) – The x-coordinate of the chip where the message should first arrive on the board
- **y** (*int*) – The y-coordinate of the chip where the message should first arrive on the board
- **multicast_message** (spinnman.messages.multicast_message.MulticastMessage) – A multicast message to send
- **connection** (spinnman.connections.abstract_multicast_sender.AbstractMulticastSe – A specific connection over which to send the message. If not specified, an appropriate connection is chosen automatically

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanUnsupportedOperationException** –
  - If there is no connection that supports sending over multicast (or the given connection does not)
  - If there is no connection that can make the packet arrive at the selected chip (ignoring routing tables)

send_scp_message(*message*, *retry_codes=(<SCPResult.RC_P2P_TIMEOUT: 142>, <SCPResult.RC_TIMEOUT: 134>, <SCPResult.RC_LEN: 129>)*, *n_retries=10*, *timeout=1*, *connection=None*)
Sends an SCP message, and gets a response

Parameters

- **message** (`spinnman.messages.scp.abstract_scp_request.AbstractSCPRequest`) – The message to send
- **retry_codes** (iterable of `spinnman.messages.scp.scp_result.SCPResult`) – The response codes which will result in a retry if received as a response
- **n_retries** (*int*) – The number of times to retry when a retry code is received
- **timeout** (*int*) – The timeout to use when receiving a response
- **connection** (`spinnman.connections.abstract_connection.AbstractConnection`) – The connection to use

Returns The received response, or the callback if get_callback is True

Return type `spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse`

Raises

- **spinnman.exceptions.SpinnmanTimeoutException** – If there is a timeout before a message is received
- **spinnman.exceptions.SpinnmanInvalidParameterException** – If one of the fields of the received message is invalid
- **spinnman.exceptions.SpinnmanInvalidPacketException** –
  - If the message is not a recognized packet type
  - If a packet is received that is not a valid response
- **spinnman.exceptions.SpinnmanUnsupportedOperationException** – If no connection can send the type of message given
- **spinnman.exceptions.SpinnmanIOException** – If there is an error sending the message or receiving the response
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If the response is not one of the expected codes

send_sdp_message (*message*, *connection=None*)

Sends a EIEIO command message using one of the connections.

Parameters

- **message** (*SDPMessage*) – The message to send
- **connection** (`spinnman.connections.abstract_connection.AbstractConnection`) – An optional connection to use

Returns None

send_signal (*app_id*, *signal*)

Send a signal to an application

Parameters

- **app_id** (*int*) – The id of the application to send to
- **signal** (`spinnman.messages.scp.scp_signal.SCPSignal` :py:class:'spinnman.messages.scp.scp_signal.SCPSignal') – The signal to send

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If signal is not a valid signal
  - If app_id is not a valid application id
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

set_ip_tag(*ip_tag*)

Set up an ip tag

Parameters **ip_tag** (`spinn_machine.tags.iptag.IPTag`) – The tag to set up; note board_address can be None, in which case, the tag will be assigned to all boards

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If the ip tag fields are incorrect
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

set_leds(*x*, *y*, *cpu*, *led_states*)

Set LED states. :param x: The x-coordinate of the chip on which to set the LEDs :type x: int :param y: The x-coordinate of the chip on which to set the LEDs :type y: int :param cpu: The CPU of the chip on which to set the LEDs :type cpu: int :param led_states: A dictionary mapping LED index to state with 0 being

off, 1 on and 2 inverted.

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format

- **spinnman.exceptions.SpinnmanInvalidParameterException** – If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

set_reverse_ip_tag(*reverse_ip_tag*)

Set up a reverse ip tag

Parameters **reverse_ip_tag** (spinn_machine.tags.reverse_ip_tag.ReverseIPTag) – The reverse tag to set up; note board_address can be None, in which case, the tag will be assigned to all boards

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If the reverse ip tag fields are incorrect
  - If a packet is received that has invalid parameters
  - If the UDP port is one that is already used by spiNNaker for system functions
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

set_router_diagnostic_filter(*x*, *y*, *position*, *diagnostic_filter*)

Sets a router diagnostic filter in a router

Parameters

- **x** (*int*) – the x address of the router in which this filter is being set
- **y** (*int*) – the y address of the router in which this filter is being set
- **position** (*int*) – the position in the list of filters where this filter is to be added
- **diagnostic_filter** (spinnman.model.diagnostic_filter.DiagnosticFilter) – the diagnostic filter being set in the placed, between 0 and 15 (note that positions 0 to 11 are used by the default filters, and setting these positions will result in a warning).

Returns None

Raises

- **spinnman.exceptions.SpinnmanIOException** –
  - If there is an error communicating with the board
  - If there is an error reading the data
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If x, y does not lead to a valid chip

- If position is less than 0 or more than 15

- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

stop_application(*app_id*)
Sends a stop request for an app_id

Parameters **app_id** (*int*) – The id of the application to send to

Raises

- **spinnman.exceptions.SpinnmanIOException** – If there is an error communicating with the board
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If app_id is not a valid application id
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

write_memory(*x*, *y*, *base_address*, *data*, *n_bytes=None*)
Write to the SDRAM on the board

Parameters

- **x** (*int*) – The x-coordinate of the chip where the memory is to be written to
- **y** (*int*) – The y-coordinate of the chip where the memory is to be written to
- **base_address** (*int*) – The address in SDRAM where the region of memory is to be written
- **data** (spinnman.data.abstract_data_reader.AbstractDataReader or bytearray or int) – The data to write. Should be one of the following: * An instance of AbstractDataReader * A bytearray * A single integer - will be written using little-endian byte ordering
- **n_bytes** (*int*) – The amount of data to be written in bytes. If not specified: * If data is an AbstractDataReader, an error is raised * If data is a bytearray, the length of the bytearray will be used * If data is an int, 4 will be used

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** –
  - If there is an error communicating with the board
  - If there is an error reading the data
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If x, y does not lead to a valid chip
  - If a packet is received that has invalid parameters

- If base_address is not a positive integer
- If data is an AbstractDataReader but n_bytes is not specified
- If data is an int and n_bytes is more than 4
- If n_bytes is less than 0

- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

write_memory_flood(*base_address*, *data*, *n_bytes=None*)

Write to the SDRAM of all chips.

Parameters

- **base_address** (*int*) – The address in SDRAM where the region of memory is to be written
- **data** (spinnman.data.abstract_data_reader.AbstractDataReader or bytearray or int) – The data that is to be written. Should be one of the following: * An instance of AbstractDataReader * A bytearray * A single integer
- **n_bytes** (*int*) – The amount of data to be written in bytes. If not specified: * If data is an AbstractDataReader, an error is raised * If data is a bytearray, the length of the bytearray will be used * If data is an int, 4 will be used * If n_bytes is less than 0

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** –
  - If there is an error communicating with the board
  - If there is an error reading the executable
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If one of the specified chips is not valid
  - If app_id is an invalid application id
  - If a packet is received that has invalid parameters
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

write_neighbour_memory(*x*, *y*, *cpu*, *link*, *base_address*, *data*, *n_bytes=None*)

Write to the memory of a neighbouring chip using a LINK_READ SCP command. If sent to a BMP, this command can be used to communicate with the FPGAs' debug registers.

Parameters

- **x** (*int*) – The x-coordinate of the chip whose neighbour is to be written to
- **y** (*int*) – The y-coordinate of the chip whose neighbour is to be written to
- **cpu** (*int*) – The cpu to use, typically 0 (or if a BMP, the slot number)
- **link** – The link index to send the request to (or if BMP, the FPGA

number) :type link: int :param base_address: The address in SDRAM where the region of memory is to be written :type base_address: int :param data: The data to write. Should be one of the following:

•An instance of AbstractDataReader

•A bytearray

•A single integer - will be written using little-endian byte ordering

Parameters **n_bytes** (*int*) – The amount of data to be written in bytes. If not specified: * If data is an AbstractDataReader, an error is raised * If data is a bytearray, the length of the bytearray will be used * If data is an int, 4 will be used

Returns Nothing is returned

Return type None

Raises

- **spinnman.exceptions.SpinnmanIOException** –
  - If there is an error communicating with the board
  - If there is an error reading the data
- **spinnman.exceptions.SpinnmanInvalidPacketException** – If a packet is received that is not in the valid format
- **spinnman.exceptions.SpinnmanInvalidParameterException** –
  - If x, y does not lead to a valid chip
  - If a packet is received that has invalid parameters
  - If base_address is not a positive integer
  - If data is an AbstractDataReader but n_bytes is not specified
  - If data is an int and n_bytes is more than 4
  - If n_bytes is less than 0
- **spinnman.exceptions.SpinnmanUnexpectedResponseCodeException** – If a response indicates an error during the exchange

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*

S

A

E

F

H

I

L

M

P

R

S

T

U

V

W