
SpiNNFrontEndCommonEnd Documentation

Jan 29, 2020

Contents

1	spinn_front_end_common package	3
1.1	Subpackages	3
1.1.1	spinn_front_end_common.abstract_models package	3
1.1.1.1	Subpackages	3
1.1.1.2	Submodules	5
1.1.1.3	spinn_front_end_common.abstract_models.abstract_can_reset module	5
1.1.1.4	spinn_front_end_common.abstract_models.abstract_changable_after_run module	5
1.1.1.5	spinn_front_end_common.abstract_models.abstract_generates_data_specification module	6
1.1.1.6	spinn_front_end_common.abstract_models.abstract_has_associated_binary module	6
1.1.1.7	spinn_front_end_common.abstract_models.abstract_machine_allocation_controller module	6
1.1.1.8	spinn_front_end_common.abstract_models.abstract_provides_incoming_partition_constraints module	7
1.1.1.9	spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping module	7
1.1.1.10	spinn_front_end_common.abstract_models.abstract_provides_n_keys_for_partition module	7
1.1.1.11	spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints module	8
1.1.1.12	spinn_front_end_common.abstract_models.abstract_recordable module	8
1.1.1.13	spinn_front_end_common.abstract_models.abstract_rewrites_data_specification module	8
1.1.1.14	spinn_front_end_common.abstract_models.abstract_send_me_multicast_commands_vertex module	8
1.1.1.15	spinn_front_end_common.abstract_models.abstract_supports_database_injection module	9
1.1.1.16	spinn_front_end_common.abstract_models.abstract_uses_memory_io module	9
1.1.1.17	spinn_front_end_common.abstract_models.abstract_vertex_with_dependent_vertices module	9
1.1.1.18	Module contents	10
1.1.2	spinn_front_end_common.common_model_binaries package	13
1.1.2.1	Module contents	13
1.1.3	spinn_front_end_common.interface package	13
1.1.3.1	Subpackages	13
1.1.3.2	Submodules	52

1.1.3.3	spinn_front_end_common.interface.abstract_spinnaker_base module	52
1.1.3.4	spinn_front_end_common.interface.config_handler module	52
1.1.3.5	spinn_front_end_common.interface.java_caller module	52
1.1.3.6	spinn_front_end_common.interface.simulator_state module	54
1.1.3.7	Module contents	54
1.1.4	spinn_front_end_common.mapping_algorithms package	54
1.1.4.1	Subpackages	54
1.1.4.2	Module contents	54
1.1.5	spinn_front_end_common.utilities package	54
1.1.5.1	Subpackages	54
1.1.5.2	Submodules	96
1.1.5.3	spinn_front_end_common.utilities.constants module	96
1.1.5.4	spinn_front_end_common.utilities.exceptions module	96
1.1.5.5	spinn_front_end_common.utilities.failed_state module	97
1.1.5.6	spinn_front_end_common.utilities.function_list module	97
1.1.5.7	spinn_front_end_common.utilities.globals_variables module	97
1.1.5.8	spinn_front_end_common.utilities.helpful_functions module	98
1.1.5.9	spinn_front_end_common.utilities.math_constants module	101
1.1.5.10	spinn_front_end_common.utilities.simulator_interface module	101
1.1.5.11	Module contents	102
1.1.6	spinn_front_end_common.utility_models package	103
1.1.6.1	Submodules	103
1.1.6.2	spinn_front_end_common.utility_models.chip_power_monitor module	103
1.1.6.3	spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex module	104
1.1.6.4	spinn_front_end_common.utility_models.command_sender module	106
1.1.6.5	spinn_front_end_common.utility_models.command_sender_machine_vertex module	107
1.1.6.6	spinn_front_end_common.utility_models.data_speed_up_packet_gatherer module	109
1.1.6.7	spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex module	110
1.1.6.8	spinn_front_end_common.utility_models.extra_monitor_support module	114
1.1.6.9	spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex module	115
1.1.6.10	spinn_front_end_common.utility_models.live_packet_gather module	118
1.1.6.11	spinn_front_end_common.utility_models.live_packet_gather_machine_vertex module	119
1.1.6.12	spinn_front_end_common.utility_models.multi_cast_command module	121
1.1.6.13	spinn_front_end_common.utility_models.reverse_ip_tag_multi_cast_source module	122
1.1.6.14	spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex module	124
1.1.6.15	Module contents	126
1.2	Module contents	148
2	Indices and tables	149
	Python Module Index	151
	Index	155

These pages document the python code for the [SpiNNFrontEndCommon](#) module which is part of the [SpiNNaker Project](#).

This code depends on [SpiNNUtils](#), [SpiNNMachine](#), [SpiNNStorageHandlers](#), [SpiNNMan](#), [PACMAN](#), [DataSpecification](#) ([Combined_documentation](#)).

Contents:

spinn_front_end_common package

1.1 Subpackages

1.1.1 spinn_front_end_common.abstract_models package

1.1.1.1 Subpackages

spinn_front_end_common.abstract_models.impl package

Submodules

spinn_front_end_common.abstract_models.impl.machine_allocation_controller module

```
class spinn_front_end_common.abstract_models.impl.machine_allocation_controller.MachineAllocationController
    Bases: spinn_front_end_common.abstract_models.abstract_machine_allocation_controller.AbstractMachineAllocationController

    close()
        Indicate that the use of the machine is complete
```

spinn_front_end_common.abstract_models.impl.machine_data_specable_vertex module

```
class spinn_front_end_common.abstract_models.impl.machine_data_specable_vertex.MachineDataSpecableVertex
    Bases: spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification

    generate_data_specification(*args, **kwargs)
        Generate a data specification.

        Parameters
```

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

generate_machine_data_specification (*spec, placement, machine_graph, routing_info, iptags, reverse_iptags, machine_time_step, time_scale_factor*)

Parameters

- **spec** (*DataSpecificationGenerator*) – The data specification to write into.
- **placement** – Where this node is on the SpiNNaker machine.
- **machine_graph** – The graph containing this node.
- **routing_info** –
- **iptags** –
- **reverse_iptags** –
- **machine_time_step** –
- **time_step_factor** –

spinn_front_end_common.abstract_models.impl.provides_key_to_atom_mapping_impl module

class `spinn_front_end_common.abstract_models.impl.provides_key_to_atom_mapping_impl`.**Provide**

Bases: `spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping`.
AbstractProvidesKeyToAtomMapping

routing_key_partition_atom_mapping (**args, **kwargs*)

Returns a list of atom to key mapping.

Parameters

- **routing_info** – the routing info object to consider
- **partition** – the routing partition to handle.

Returns a iterable of tuples of atom IDs to keys.

Module contents

class `spinn_front_end_common.abstract_models.impl`.**MachineAllocationController** (*thread_name*)

Bases: `spinn_front_end_common.abstract_models.abstract_machine_allocation_controller`.
AbstractMachineAllocationController

close ()

Indicate that the use of the machine is complete

class `spinn_front_end_common.abstract_models.impl`.**MachineDataSpecableVertex** (**args, **kwargs*)

Bases: `spinn_front_end_common.abstract_models.abstract_generates_data_specification`.
AbstractGeneratesDataSpecification

generate_data_specification (**args, **kwargs*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

generate_machine_data_specification (*spec, placement, machine_graph, routing_info, iptags, reverse_iptags, machine_time_step, time_scale_factor*)

Parameters

- **spec** (*DataSpecificationGenerator*) – The data specification to write into.
- **placement** – Where this node is on the SpiNNaker machine.
- **machine_graph** – The graph containing this node.
- **routing_info** –
- **iptags** –
- **reverse_iptags** –
- **machine_time_step** –
- **time_step_factor** –

class `spinn_front_end_common.abstract_models.impl.ProvidesKeyToAtomMappingImpl`

Bases: `spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping.AbstractProvidesKeyToAtomMapping`

routing_key_partition_atom_mapping (**args, **kwargs*)

Returns a list of atom to key mapping.

Parameters

- **routing_info** – the routing info object to consider
- **partition** – the routing partition to handle.

Returns a iterable of tuples of atom IDs to keys.

1.1.1.2 Submodules

1.1.1.3 `spinn_front_end_common.abstract_models.abstract_can_reset` module

class `spinn_front_end_common.abstract_models.abstract_can_reset.AbstractCanReset`

Bases: `object`

Indicates an object that can be reset to time 0

reset_to_first_timestep ()

Reset the object to first time step

1.1.1.4 `spinn_front_end_common.abstract_models.abstract_changable_after_run` module

class `spinn_front_end_common.abstract_models.abstract_changable_after_run.AbstractChangableAfterRun`

Bases: `object`

An item that can be changed after a call to run, the changes to which might or might not require mapping or data generation.

mark_no_changes()

Marks the point after which changes are reported, so that new changes can be detected before the next check.

requires_data_generation

True if changes that have been made require that data generation be performed. By default this returns False but can be overridden to indicate changes that require data regeneration.

Return type bool

requires_mapping

True if changes that have been made require that mapping be performed. By default this returns False but can be overridden to indicate changes that require mapping.

Return type bool

1.1.1.5 spinn_front_end_common.abstract_models.abstract_generates_data_specification module

class spinn_front_end_common.abstract_models.abstract_generates_data_specification.**AbstractGeneratesDataSpecification**

Bases: object

generate_data_specification(*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

1.1.1.6 spinn_front_end_common.abstract_models.abstract_has_associated_binary module

class spinn_front_end_common.abstract_models.abstract_has_associated_binary.**AbstractHasAssociatedBinary**

Bases: object

Marks a machine graph vertex that can be launched on a SpiNNaker core.

get_binary_file_name()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()

Get the start type of the binary to be run.

Return type *ExecutableType*

1.1.1.7 spinn_front_end_common.abstract_models.abstract_machine_allocation_controller module

class spinn_front_end_common.abstract_models.abstract_machine_allocation_controller.**AbstractMachineAllocationController**

Bases: object

An object that controls the allocation of a machine

close()

Indicate that the use of the machine is complete

extend_allocation (*new_total_run_time*)

Extend the allocation of the machine from the original run time.

Parameters *new_total_run_time* – The total run time that is now required starting from when the machine was first allocated

1.1.1.8 `spinn_front_end_common.abstract_models.abstract_provides_incoming_partition_constraints` module

class `spinn_front_end_common.abstract_models.abstract_provides_incoming_partition_constraints`
Bases: `object`

A vertex that can provide constraints for its incoming edge partitions.

get_incoming_partition_constraints (*partition*)

Get constraints to be added to the given edge that goes in to a vertex of this vertex.

Parameters *partition* (`OutgoingPartition`) – An partition that goes in to this vertex

Returns A list of constraints

Return type `list(AbstractConstraint)`

1.1.1.9 `spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping` module

class `spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping`.**Abstract**
Bases: `object`

Interface to provide a mapping between routing key partitions and atom IDs

routing_key_partition_atom_mapping (*routing_info*, *partition*)

Returns a list of atom to key mapping.

Parameters

- **routing_info** – the routing info object to consider
- **partition** – the routing partition to handle.

Returns a iterable of tuples of atom IDs to keys.

1.1.1.10 `spinn_front_end_common.abstract_models.abstract_provides_n_keys_for_partition` module

class `spinn_front_end_common.abstract_models.abstract_provides_n_keys_for_partition`.**Abstract**
Bases: `object`

Allows a vertex to provide the number of keys for a partition of edges, rather than relying on the number of atoms in the pre-vertex.

get_n_keys_for_partition (*partition*, *graph_mapper*)

Get the number of keys required by the given partition of edges.

Parameters

- **partition** (`OutgoingPartition`) – An partition that comes out of this vertex
- **graph_mapper** (`GraphMapper`) – A mapper between the graphs

Returns A list of constraints

Return type list(*AbstractConstraint*)

1.1.1.11 spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints module

class spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints

Bases: object

A vertex that can provide constraints for its outgoing edge partitions.

get_outgoing_partition_constraints (*partition*)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters *partition* – An edge that comes out of this vertex

Returns A list of constraints

Return type list(*AbstractConstraint*)

1.1.1.12 spinn_front_end_common.abstract_models.abstract_recordable module

class spinn_front_end_common.abstract_models.abstract_recordable.**AbstractRecordable**

Bases: object

Indicates that an object might record some data in to SDRAM

is_recording ()

Deduce if the recorder is actually recording

1.1.1.13 spinn_front_end_common.abstract_models.abstract_rewrites_data_specification module

class spinn_front_end_common.abstract_models.abstract_rewrites_data_specification.**AbstractRewritesDataSpecification**

Bases: object

Indicates an object that allows data to be changed after run, and so can rewrite the data specification

mark_regions_reloaded ()

Indicate that the regions have been reloaded

regenerate_data_specification (*spec, placement*)

Regenerate the data specification, only generating regions that have changed and need to be reloaded

requires_memory_regions_to_be_reloaded ()

Return true if any data region needs to be reloaded

Return type bool

1.1.1.14 spinn_front_end_common.abstract_models.abstract_send_me_multicast_commands_vertex module

class spinn_front_end_common.abstract_models.abstract_send_me_multicast_commands_vertex.**AbstractSendMeMulticastCommandsVertex**

Bases: object

A vertex which wants to commands to be sent to it as multicast packets at fixed points in the simulation

pause_stop_commands

The commands needed when pausing or stopping simulation

start_resume_commands

The commands needed when starting or resuming simulation

timed_commands

The commands to be sent at given times in the simulation

1.1.1.15 spinn_front_end_common.abstract_models.abstract_supports_database_injection module

class spinn_front_end_common.abstract_models.abstract_supports_database_injection.**AbstractSupportsDatabaseInjection**

Bases: object

Marks a machine vertex as supporting injection of information via a database running on the controlling host.

is_in_injection_mode

Whether this vertex is actually in injection mode.

1.1.1.16 spinn_front_end_common.abstract_models.abstract_uses_memory_io module

class spinn_front_end_common.abstract_models.abstract_uses_memory_io.**AbstractUsesMemoryIO**

Bases: object

Indicates that the class will write data using the MemoryIO interface

get_memory_io_data_size()

Get the size of the data area to allocate to this vertex

Returns The size of the data area in bytes

Return type int

write_data_to_memory_io(memory, tag)

Write the data to the given memory object

Parameters

- **memory** (`MemoryIO`) – The memory to write to
- **tag** (`int`) – The tag given to the allocated memory

1.1.1.17 spinn_front_end_common.abstract_models.abstract_vertex_with_dependent_vertices module

class spinn_front_end_common.abstract_models.abstract_vertex_with_dependent_vertices.**AbstractVertexWithDependentVertices**

Bases: object

A vertex with a dependent vertices, which should be connected to this vertex by an edge directly to each of them

dependent_vertices()

Return the vertices which this vertex depends upon

edge_partition_identifiers_for_dependent_vertex(vertex)

Return the dependent edge identifiers for this vertex

1.1.1.18 Module contents

class spinn_front_end_common.abstract_models.**AbstractChangableAfterRun**

Bases: object

An item that can be changed after a call to run, the changes to which might or might not require mapping or data generation.

mark_no_changes ()

Marks the point after which changes are reported, so that new changes can be detected before the next check.

requires_data_generation

True if changes that have been made require that data generation be performed. By default this returns False but can be overridden to indicate changes that require data regeneration.

Return type bool

requires_mapping

True if changes that have been made require that mapping be performed. By default this returns False but can be overridden to indicate changes that require mapping.

Return type bool

class spinn_front_end_common.abstract_models.**AbstractGeneratesDataSpecification**

Bases: object

generate_data_specification (*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

class spinn_front_end_common.abstract_models.**AbstractHasAssociatedBinary**

Bases: object

Marks a machine graph vertex that can be launched on a SpiNNaker core.

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType*

class spinn_front_end_common.abstract_models.**AbstractMachineAllocationController**

Bases: object

An object that controls the allocation of a machine

close ()

Indicate that the use of the machine is complete

extend_allocation (*new_total_run_time*)

Extend the allocation of the machine from the original run time.

Parameters `new_total_run_time` – The total run time that is now required starting from when the machine was first allocated

class `spinn_front_end_common.abstract_models.AbstractProvidesIncomingPartitionConstraints`
Bases: `object`

A vertex that can provide constraints for its incoming edge partitions.

get_incoming_partition_constraints (*partition*)

Get constraints to be added to the given edge that goes in to a vertex of this vertex.

Parameters `partition` (`OutgoingPartition`) – An partition that goes in to this vertex

Returns A list of constraints

Return type `list(AbstractConstraint)`

class `spinn_front_end_common.abstract_models.AbstractProvidesKeyToAtomMapping`
Bases: `object`

Interface to provide a mapping between routing key partitions and atom IDs

routing_key_partition_atom_mapping (*routing_info*, *partition*)

Returns a list of atom to key mapping.

Parameters

- **routing_info** – the routing info object to consider
- **partition** – the routing partition to handle.

Returns a iterable of tuples of atom IDs to keys.

class `spinn_front_end_common.abstract_models.AbstractProvidesNKeysForPartition`
Bases: `object`

Allows a vertex to provide the number of keys for a partition of edges, rather than relying on the number of atoms in the pre-vertex.

get_n_keys_for_partition (*partition*, *graph_mapper*)

Get the number of keys required by the given partition of edges.

Parameters

- **partition** (`OutgoingPartition`) – An partition that comes out of this vertex
- **graph_mapper** (`GraphMapper`) – A mapper between the graphs

Returns A list of constraints

Return type `list(AbstractConstraint)`

class `spinn_front_end_common.abstract_models.AbstractProvidesOutgoingPartitionConstraints`
Bases: `object`

A vertex that can provide constraints for its outgoing edge partitions.

get_outgoing_partition_constraints (*partition*)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters `partition` – An edge that comes out of this vertex

Returns A list of constraints

Return type `list(AbstractConstraint)`

class spinn_front_end_common.abstract_models.**AbstractRecordable**

Bases: object

Indicates that an object might record some data in to SDRAM

is_recording()

Deduce if the recorder is actually recording

class spinn_front_end_common.abstract_models.**AbstractRewritesDataSpecification**

Bases: object

Indicates an object that allows data to be changed after run, and so can rewrite the data specification

mark_regions_reloaded()

Indicate that the regions have been reloaded

regenerate_data_specification (*spec, placement*)

Regenerate the data specification, only generating regions that have changed and need to be reloaded

requires_memory_regions_to_be_reloaded()

Return true if any data region needs to be reloaded

Return type bool

class spinn_front_end_common.abstract_models.**AbstractSendMeMulticastCommandsVertex**

Bases: object

A vertex which wants to commands to be sent to it as multicast packets at fixed points in the simulation

pause_stop_commands

The commands needed when pausing or stopping simulation

start_resume_commands

The commands needed when starting or resuming simulation

timed_commands

The commands to be sent at given times in the simulation

class spinn_front_end_common.abstract_models.**AbstractSupportsDatabaseInjection**

Bases: object

Marks a machine vertex as supporting injection of information via a database running on the controlling host.

is_in_injection_mode

Whether this vertex is actually in injection mode.

class spinn_front_end_common.abstract_models.**AbstractVertexWithEdgeToDependentVertices**

Bases: object

A vertex with a dependent vertices, which should be connected to this vertex by an edge directly to each of them

dependent_vertices()

Return the vertices which this vertex depends upon

edge_partition_identifiers_for_dependent_vertex (*vertex*)

Return the dependent edge identifiers for this vertex

class spinn_front_end_common.abstract_models.**AbstractUsesMemoryIO**

Bases: object

Indicates that the class will write data using the MemoryIO interface

get_memory_io_data_size()

Get the size of the data area to allocate to this vertex

Returns The size of the data area in bytes

Return type int

write_data_to_memory_io (*memory*, *tag*)

Write the data to the given memory object

Parameters

- **memory** (*MemoryIO*) – The memory to write to
- **tag** (*int*) – The tag given to the allocated memory

class spinn_front_end_common.abstract_models.**AbstractCanReset**

Bases: object

Indicates an object that can be reset to time 0

reset_to_first_timestep ()

Reset the object to first time step

1.1.2 spinn_front_end_common.common_model_binaries package

1.1.2.1 Module contents

This module contains no python code

1.1.3 spinn_front_end_common.interface package

1.1.3.1 Subpackages

spinn_front_end_common.interface.buffer_management package

Subpackages

spinn_front_end_common.interface.buffer_management.buffer_models package

Submodules

spinn_front_end_common.interface.buffer_management.buffer_models.abstract_receive_buffers_to_host module

class spinn_front_end_common.interface.buffer_management.buffer_models.abstract_receive_buffers_to_host

Bases: object

Indicates that this object can receive buffers

get_recorded_region_ids ()

Get the recording region IDs that have been recorded using buffering

Returns The region numbers that have active recording

Return type iterable(int)

get_recording_region_base_address (*txrx*, *placement*)

Get the recording region base address

Parameters

- **txrx** – the SpiNNMan instance
- **placement** – the placement object of the core to find the address of

Returns the base address of the recording region

`spinn_front_end_common.interface.buffer_management.buffer_models.abstract_sends_buffers_from_host` module

class `spinn_front_end_common.interface.buffer_management.buffer_models.abstract_sends_buffers_from_host`

Bases: `object`

Interface to an object that sends buffers of keys to be transmitted at given timestamps in the simulation

buffering_input ()

Return True if the input of this vertex is to be buffered

get_next_key (*region*)

Get the next key in the given region

Parameters **region** (*int*) – The region to get the next key from

Returns The next key, or None if there are no more keys

Return type `int`

get_next_timestamp (*region*)

Get the next timestamp at which there are still keys to be sent for the given region

Parameters **region** (*int*) – The region to get the timestamp for

Returns The timestamp of the next available keys

Return type `int`

get_region_buffer_size (*region*)

Get the size of the buffer to be used in SDRAM on the machine for the region in bytes

Parameters **region** (*int*) – The region to get the buffer size of

Returns The size of the buffer space in bytes

Return type `int`

get_regions ()

Get the set of regions for which there are keys to be sent

Returns Iterable of region IDs

Return type `iterable(int)`

is_empty (*region*)

Return true if there are no spikes to be buffered for the specified region

Parameters **region** (*int*) – The region to get the next key from

Returns Whether there are no keys to send for the region

Return type `bool`

is_next_key (*region, timestamp*)

Determine if there are still keys to be sent at the given timestamp for the given region

Parameters

- **region** (*int*) – The region to determine if there are keys for
- **timestamp** (*int*) – The timestamp to determine if there are more keys for

Returns Whether there are more keys to send for the parameters

Return type bool

is_next_timestamp (*region*)

Determine if there is another timestamp with data to be sent

Parameters **region** (*int*) – The region to determine if there is more data for

Returns Whether there is more data

Return type bool

rewind (*region*)

Rewinds the internal buffer in preparation of re-sending the spikes

Parameters **region** (*int*) – The region to rewind

spinn_front_end_common.interface.buffer_management.buffer_models.sends_buffers_from_host_pre_buffered_in
module

class spinn_front_end_common.interface.buffer_management.buffer_models.sends_buffers_from_l

Bases: *spinn_front_end_common.interface.buffer_management.buffer_models.
abstract_sends_buffers_from_host.AbstractSendsBuffersFromHost*

Implementation of AbstractSendsBuffersFromHost which uses an existing set of buffers for the details

buffering_input ()

Return True if the input of this vertex is to be buffered

get_next_key (*region*)

Get the next key for a given region

Parameters **region** – the region to get the next key from

get_next_timestamp (*region*)

Return the next time stamp available in the buffered region

Parameters **region** – the region ID which is being asked

Returns the next time stamp

get_regions ()

Return the regions which has buffers to send

is_empty (*region*)

Check if a region is empty

Parameters **region** – the region ID to check

Returns bool

is_next_key (*region, timestamp*)

Check if there is more keys to transmit for a given region in a given timestamp

Parameters

- **region** – the region ID to check
- **timestamp** – the timestamp to check

Returns bool

is_next_timestamp (*region*)

Check if there are more time stamps which need transmitting

Parameters **region** – the region to check

Returns boolean

rewind (*region*)

Rewinds the internal buffer in preparation of re-sending the spikes

Parameters **region** (*int*) – The region to rewind

send_buffers

Module contents

class spinn_front_end_common.interface.buffer_management.buffer_models.**AbstractReceiveBuffer**

Bases: object

Indicates that this object can receive buffers

get_recorded_region_ids ()

Get the recording region IDs that have been recorded using buffering

Returns The region numbers that have active recording

Return type iterable(int)

get_recording_region_base_address (*txrx, placement*)

Get the recording region base address

Parameters

- **txrx** – the SpiNNMan instance
- **placement** – the placement object of the core to find the address of

Returns the base address of the recording region

class spinn_front_end_common.interface.buffer_management.buffer_models.**AbstractSendsBuffers**

Bases: object

Interface to an object that sends buffers of keys to be transmitted at given timestamps in the simulation

buffering_input ()

Return True if the input of this vertex is to be buffered

get_next_key (*region*)

Get the next key in the given region

Parameters **region** (*int*) – The region to get the next key from

Returns The next key, or None if there are no more keys

Return type int

get_next_timestamp (*region*)

Get the next timestamp at which there are still keys to be sent for the given region

Parameters **region** (*int*) – The region to get the timestamp for

Returns The timestamp of the next available keys

Return type int

get_region_buffer_size (*region*)

Get the size of the buffer to be used in SDRAM on the machine for the region in bytes

Parameters **region** (*int*) – The region to get the buffer size of

Returns The size of the buffer space in bytes

Return type *int*

get_regions ()

Get the set of regions for which there are keys to be sent

Returns Iterable of region IDs

Return type *iterable(int)*

is_empty (*region*)

Return true if there are no spikes to be buffered for the specified region

Parameters **region** (*int*) – The region to get the next key from

Returns Whether there are no keys to send for the region

Return type *bool*

is_next_key (*region, timestamp*)

Determine if there are still keys to be sent at the given timestamp for the given region

Parameters

- **region** (*int*) – The region to determine if there are keys for
- **timestamp** (*int*) – The timestamp to determine if there are more keys for

Returns Whether there are more keys to send for the parameters

Return type *bool*

is_next_timestamp (*region*)

Determine if there is another timestamp with data to be sent

Parameters **region** (*int*) – The region to determine if there is more data for

Returns Whether there is more data

Return type *bool*

rewind (*region*)

Rewinds the internal buffer in preparation of re-sending the spikes

Parameters **region** (*int*) – The region to rewind

class `spinn_front_end_common.interface.buffer_management.buffer_models.SendsBuffersFromHost`

Bases: `spinn_front_end_common.interface.buffer_management.buffer_models.abstract_sends_buffers_from_host.AbstractSendsBuffersFromHost`

Implementation of `AbstractSendsBuffersFromHost` which uses an existing set of buffers for the details

buffering_input ()

Return True if the input of this vertex is to be buffered

get_next_key (*region*)

Get the next key for a given region

Parameters **region** – the region to get the next key from

get_next_timestamp (*region*)

Return the next time stamp available in the buffered region

Parameters *region* – the region ID which is being asked

Returns the next time stamp

get_regions ()

Return the regions which has buffers to send

is_empty (*region*)

Check if a region is empty

Parameters *region* – the region ID to check

Returns bool

is_next_key (*region, timestamp*)

Check if there is more keys to transmit for a given region in a given timestamp

Parameters

- *region* – the region ID to check
- *timestamp* – the timestamp to check

Returns bool

is_next_timestamp (*region*)

Check if there are more time stamps which need transmitting

Parameters *region* – the region to check

Returns boolean

rewind (*region*)

Rewinds the internal buffer in preparation of re-sending the spikes

Parameters *region* (*int*) – The region to rewind

send_buffers

spinn_front_end_common.interface.buffer_management.storage_objects package

Submodules

spinn_front_end_common.interface.buffer_management.storage_objects.abstract_database module

class spinn_front_end_common.interface.buffer_management.storage_objects.abstract_database
Bases: object

This API separates the required database calls from the implementation.

Methods here are designed for the convenience of the caller not the database.

There should only ever be a single Database Object in use at any time. In the case of application_graph_changed the first should closed and a new one created.

Do not assume that just because 2 database objects where opened with the same parameters (for example sqllite file) that they hold the same data. In fact the second init is allowed to delete any previous data.

While not recommended implementation objects are allowed to hold data in memory, with the exception of data required by the java which must be in the database once commit is called.

clear()

Clears the data for all regions.

Warning: This method will be removed when the database moves to keeping data after reset.

Return type None

close()

Signals that the database can be closed and will not be reused.

Once this is called any other method in this API is allowed to raise any kind of exception.

get_region_data (*x, y, p, region*)

Get the data stored for a given region of a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool)

store_data_in_region_buffer (*x, y, p, region, data*)

Store some information in the correspondent buffer class for a specific chip, core and region

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored

spinn_front_end_common.interface.buffer_management.storage_objects.buffered_receiving_data module

class spinn_front_end_common.interface.buffer_management.storage_objects.buffered_receiving_data

Bases: object

Stores the information received through the buffering output technique from the SpiNNaker system. The data kept includes the last sent packet and last received packet, their correspondent sequence numbers, the data retrieved, a flag to identify if the data from a core has been flushed and the final state of the buffering output state machine

Parameters **report_folder** (*str*) – The directory to write the database used to store some of the data.

clear (*x*, *y*, *p*, *region_id*)

Clears the data from a given data region (only clears things associated with a given data recording region).

Parameters

- **x** (*int*) – placement x coordinate
- **y** (*int*) – placement y coordinate
- **p** (*int*) – placement p coordinate
- **region_id** (*int*) – the recording region ID to clear data from

Return type None

flushing_data_from_region (*x*, *y*, *p*, *region*, *data*)

Store flushed data from a region of a core on a chip, and mark it as being flushed

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored

get_end_buffering_sequence_number (*x*, *y*, *p*)

Get the last sequence number sent by the core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns The last sequence number

Return type int

get_end_buffering_state (*x*, *y*, *p*, *region*)

Get the end state of the buffering

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns The end state

get_region_data (*x*, *y*, *p*, *region*)

Get the data stored for a given region of a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool)

get_region_data_pointer ($x, y, p, region$)

It is no longer possible to get access to the data pointer.

Use get_region_data to get the data and missing flag directly.

is_data_from_region_flushed ($x, y, p, region$)

Check if the data region has been flushed

Parameters

- **x** (int) – x coordinate of the chip
- **y** (int) – y coordinate of the chip
- **p** (int) – Core within the specified chip
- **region** (int) – Region containing the data

Returns True if the region has been flushed. False otherwise

Return type bool

is_end_buffering_sequence_number_stored (x, y, p)

Determine if the last sequence number has been retrieved

Parameters

- **x** (int) – x coordinate of the chip
- **y** (int) – y coordinate of the chip
- **p** (int) – Core within the specified chip

Returns True if the number has been retrieved

Return type bool

is_end_buffering_state_recovered ($x, y, p, region$)

Determine if the end state has been stored

Parameters

- **x** (int) – x coordinate of the chip
- **y** (int) – y coordinate of the chip
- **p** (int) – Core within the specified chip

Returns True if the state has been stored

last_received_packet_from_core (x, y, p)

Get the last packet received for a given core

Parameters

- **x** (int) – x coordinate of the chip
- **y** (int) – y coordinate of the chip
- **p** (int) – Core within the specified chip

Returns SpinnakerRequestReadData packet received

Return type `spinnman.messages.eieio.command_messages.SpinnakerRequestReadData`

last_sent_packet_to_core (*x*, *y*, *p*)

Retrieve the last packet sent to a core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns last HostDataRead packet sent

Return type `spinnman.messages.eieio.command_messages.HostDataRead`

last_sequence_no_for_core (*x*, *y*, *p*)

Get the last sequence number for a core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns last sequence number used

Return type `int`

reset ()

resume ()

Resets states so that it can behave in a resumed mode

store_data_in_region_buffer (*x*, *y*, *p*, *region*, *data*)

Store some information in the correspondent buffer class for a specific chip, core and region

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored

store_end_buffering_sequence_number (*x*, *y*, *p*, *sequence*)

Store the last sequence number sent by the core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **sequence** (*int*) – The last sequence number

store_end_buffering_state (*x*, *y*, *p*, *region*, *state*)

Store the end state of buffering

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **state** – The end state

store_last_received_packet_from_core (*x, y, p, packet*)

Store the most recent packet received from SpiNNaker for a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **packet** (`spinnman.messages.eieio.command_messages.SpiNNakerRequestReadData`) – SpiNNakerRequestReadData packet received

store_last_sent_packet_to_core (*x, y, p, packet*)

Store the last packet sent to the given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **packet** (`spinnman.messages.eieio.command_messages.HostDataRead`) – last HostDataRead packet sent

update_sequence_no_for_core (*x, y, p, sequence_no*)

Set the last sequence number used

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **sequence_no** (*int*) – last sequence number used

Return type None

spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region
module

class `spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_`
Bases: `object`

A set of keys to be sent at given timestamps for a given region of data. Note that keys must be added in timestamp order or else an exception will be raised

add_key (*timestamp, key*)

Add a key to be sent at a given time

Parameters

- **timestamp** (*int*) – The time at which the key is to be sent
- **key** (*int*) – The key to send

add_keys (*timestamp, keys*)

Add a set of keys to be sent at the given time

Parameters

- **timestamp** (*int*) – The time at which the keys are to be sent
- **keys** (*iterable (int)*) – The keys to send

clear ()

Clears the buffer

current_timestamp

The current timestamp in the iterator

get_n_keys (*timestamp*)

Get the number of keys for a given timestamp

Parameters **timestamp** – the time stamp to check if there's still keys to transmit

is_next_key (*timestamp*)

Determine if there is another key for the given timestamp

Parameters **timestamp** – the time stamp to check if there's still keys to transmit

Return type bool

is_next_timestamp

Determines if the region is empty

Returns True if the region is empty, false otherwise

Return type bool

n_timestamps

The number of timestamps available

Return type int

next_key

The next key to be sent

Return type int

next_timestamp

The next timestamp of the data to be sent, or None if no more data

Return type int or None

rewind ()

Rewind the buffer to initial position.

timestamps

The timestamps for which there are keys

Return type iterable(int)

`spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region`

Get the number of bytes used by a given number of keys

Parameters **n_keys** (*int*) – The number of keys

spinn_front_end_common.interface.buffer_management.storage_objects.buffers_sent_deque module

class spinn_front_end_common.interface.buffer_management.storage_objects.buffers_sent_deque

Bases: object

A tracker of buffers sent / to send for a region

Parameters

- **region** (*int*) – The region being managed
- **sent_stop_message** (*bool*) – True if the stop message has been sent
- **n_sequences_per_transmission** (*int*) – The number of sequences allowed in each transmission set

add_message_to_send (*message*)

Add a message to send. The message is converted to a sequenced message.

Parameters **message** (spinnman.messages.eieio.abstract_messages.
AbstractEIEIOMessage) – The message to be added

is_empty ()

Determine if there are no messages

Return type int

is_full

Determine if the number of messages sent is at the limit for the sequencing system

Return type bool

messages

The messages that have been added to the set

Return type iterable(spinnman.messages.eieio.command_messages.
host_send_sequenced_data.HostSendSequencedData)

send_stop_message ()

Send a message to indicate the end of all the messages

update_last_received_sequence_number (*last_received_sequence_no*)

Updates the last received sequence number. If the sequence number is within the valid window, packets before the sequence number within the window are removed, and the last received sequence number is updated, thus moving the window for the next call. If the sequence number is not within the valid window, it is assumed to be invalid and so is ignored.

Parameters **last_received_sequence_no** (*int*) – The new sequence number

Returns True if update went ahead, False if it was ignored

Return type bool

**spinn_front_end_common.interface.buffer_management.storage_objects.channel_buffer_state
module****class** spinn_front_end_common.interface.buffer_management.storage_objects.channel_buffer_state

Bases: object

Stores information related to a single channel output buffering state, as it is retrieved at the end of a simulation on the SpiNNaker system.

Parameters

- **start_address** – start buffering area memory address (32 bits)
- **current_write** – address where data was last written (32 bits)
- **current_read** – address where data was last read (32 bits)
- **end_address** – The address of first byte after the buffer (32 bits)
- **region_id** – The ID of the region (8 bits)
- **missing_info** – True if the region overflowed during the simulation (8 bits)
- **last_buffer_operation** – Last operation performed on the buffer - read or write (8 bits)

ChannelBufferSize = 24**static** create_from_bytearray(*data*)**current_read****current_write****end_address****is_state_updated****last_buffer_operation****missing_info****region_id****set_update_completed()****static** size_of_channel_state()**start_address****update_last_operation**(*operation*)**update_read_pointer**(*read_ptr*)

spinn_front_end_common.interface.buffer_management.storage_objects.end_buffering_state module

class spinn_front_end_common.interface.buffer_management.storage_objects.end_buffering_state

Bases: object

Stores the buffering state at the end of a simulation.

Parameters

- **buffering_out_fsm_state** – Final sequence number received
- **list_channel_buffer_state** – a list of channel state, where each channel is stored in a ChannelBufferState object

buffering_out_fsm_state

channel_buffer_state (*i*)

get_missing_info_for_region (*region_id*)

get_state_for_region (*region_id*)

static size_of_region (*n_regions_to_record*)

spinn_front_end_common.interface.buffer_management.storage_objects.sqlite_database module

class spinn_front_end_common.interface.buffer_management.storage_objects.sqlite_database

Bases: *spinn_front_end_common.interface.buffer_management.storage_objects.abstract_database.AbstractDatabase*

Specific implementation of the Database for SQLite

Parameters **database_file** (*str*) – The name of a file that contains (or will contain) an SQLite database holding the data.

clear ()

Clears the data for all regions.

Warning: This method will be removed when the database moves to keeping data after reset.

Return type None

close ()

Signals that the database can be closed and will not be reused.

Once this is called any other method in this API is allowed to raise any kind of exception.

get_region_data (*x, y, p, region*)

Get the data stored for a given region of a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data
- **x** – x coordinate of the chip

- **y** – y coordinate of the chip
- **p** – Core within the specified chip
- **region** – Region containing the data

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool) Get the data stored for a given region of a given core

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool)

store_data_in_region_buffer (*x, y, p, region, data*)

Store some information in the correspondent buffer class for a specific chip, core and region

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored
- **x** – x coordinate of the chip
- **y** – y coordinate of the chip
- **p** – Core within the specified chip
- **region** – Region containing the data to be stored
- **data** – data to be stored

Module contents

class spinn_front_end_common.interface.buffer_management.storage_objects.**AbstractDatabase**

Bases: object

This API separates the required database calls from the implementation.

Methods here are designed for the convenience of the caller not the database.

There should only ever be a single Database Object in use at any time. In the case of application_graph_changed the first should closed and a new one created.

Do not assume that just because 2 database objects where opened with the same parameters (for example sqllite file) that they hold the same data. In fact the second init is allowed to delete any previous data.

While not recommended implementation objects are allowed to hold data in memory, with the exception of data required by the java which must be in the database once commit is called.

clear ()

Clears the data for all regions.

Warning: This method will be removed when the database moves to keeping data after reset.

Return type None

close()

Signals that the database can be closed and will not be reused.

Once this is called any other method in this API is allowed to raise any kind of exception.

get_region_data (*x, y, p, region*)

Get the data stored for a given region of a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool)

store_data_in_region_buffer (*x, y, p, region, data*)

Store some information in the correspondent buffer class for a specific chip, core and region

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored

class spinn_front_end_common.interface.buffer_management.storage_objects.**BufferedReceiving**

Bases: object

Stores the information received through the buffering output technique from the SpiNNaker system. The data kept includes the last sent packet and last received packet, their correspondent sequence numbers, the data retrieved, a flag to identify if the data from a core has been flushed and the final state of the buffering output state machine

Parameters **report_folder** (*str*) – The directory to write the database used to store some of the data.

clear (*x, y, p, region_id*)

Clears the data from a given data region (only clears things associated with a given data recording region).

Parameters

- **x** (*int*) – placement x coordinate
- **y** (*int*) – placement y coordinate
- **p** (*int*) – placement p coordinate
- **region_id** (*int*) – the recording region ID to clear data from

Return type None

flushing_data_from_region (*x, y, p, region, data*)

Store flushed data from a region of a core on a chip, and mark it as being flushed

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored

get_end_buffering_sequence_number (*x, y, p*)

Get the last sequence number sent by the core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns The last sequence number

Return type int

get_end_buffering_state (*x, y, p, region*)

Get the end state of the buffering

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns The end state

get_region_data (*x, y, p, region*)

Get the data stored for a given region of a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool)

get_region_data_pointer (*x, y, p, region*)

It is no longer possible to get access to the data pointer.

Use `get_region_data` to get the data and missing flag directly.

is_data_from_region_flushed (*x, y, p, region*)

Check if the data region has been flushed

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip

- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data

Returns True if the region has been flushed. False otherwise

Return type bool

is_end_buffering_sequence_number_stored (*x, y, p*)

Determine if the last sequence number has been retrieved

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns True if the number has been retrieved

Return type bool

is_end_buffering_state_recovered (*x, y, p, region*)

Determine if the end state has been stored

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns True if the state has been stored

last_received_packet_from_core (*x, y, p*)

Get the last packet received for a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns SpinnakerRequestReadData packet received

Return type `spinnman.messages.eieio.command_messages.SpinnakerRequestReadData`

last_sent_packet_to_core (*x, y, p*)

Retrieve the last packet sent to a core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns last HostDataRead packet sent

Return type `spinnman.messages.eieio.command_messages.HostDataRead`

last_sequence_no_for_core (*x, y, p*)

Get the last sequence number for a core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

Returns last sequence number used

Return type int

reset ()

resume ()

Resets states so that it can behave in a resumed mode

store_data_in_region_buffer (*x, y, p, region, data*)

Store some information in the correspondent buffer class for a specific chip, core and region

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored

store_end_buffering_sequence_number (*x, y, p, sequence*)

Store the last sequence number sent by the core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **sequence** (*int*) – The last sequence number

store_end_buffering_state (*x, y, p, region, state*)

Store the end state of buffering

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **state** – The end state

store_last_received_packet_from_core (*x, y, p, packet*)

Store the most recent packet received from SpiNNaker for a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip

- **packet** (`spinnman.messages.eieio.command_messages.SpinnakerRequestReadData`) – SpinnakerRequestReadData packet received

store_last_sent_packet_to_core (*x, y, p, packet*)

Store the last packet sent to the given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **packet** (`spinnman.messages.eieio.command_messages.HostDataRead`) – last HostDataRead packet sent

update_sequence_no_for_core (*x, y, p, sequence_no*)

Set the last sequence number used

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **sequence_no** (*int*) – last sequence number used

Return type None

class `spinn_front_end_common.interface.buffer_management.storage_objects.BufferedSendingRe`

Bases: object

A set of keys to be sent at given timestamps for a given region of data. Note that keys must be added in timestamp order or else an exception will be raised

add_key (*timestamp, key*)

Add a key to be sent at a given time

Parameters

- **timestamp** (*int*) – The time at which the key is to be sent
- **key** (*int*) – The key to send

add_keys (*timestamp, keys*)

Add a set of keys to be sent at the given time

Parameters

- **timestamp** (*int*) – The time at which the keys are to be sent
- **keys** (*iterable(int)*) – The keys to send

clear ()

Clears the buffer

current_timestamp

The current timestamp in the iterator

get_n_keys (*timestamp*)

Get the number of keys for a given timestamp

Parameters **timestamp** – the time stamp to check if there's still keys to transmit

is_next_key (*timestamp*)

Determine if there is another key for the given timestamp

Parameters **timestamp** – the time stamp to check if there’s still keys to transmit

Return type bool

is_next_timestamp

Determines if the region is empty

Returns True if the region is empty, false otherwise

Return type bool

n_timestamps

The number of timestamps available

Return type int

next_key

The next key to be sent

Return type int

next_timestamp

The next timestamp of the data to be sent, or None if no more data

Return type int or None

rewind()

Rewind the buffer to initial position.

timestamps

The timestamps for which there are keys

Return type iterable(int)

class spinn_front_end_common.interface.buffer_management.storage_objects.**BuffersSentDeque** (*n*)

Bases: object

A tracker of buffers sent / to send for a region

Parameters

- **region** (*int*) – The region being managed
- **sent_stop_message** (*bool*) – True if the stop message has been sent
- **n_sequences_per_tranmission** (*int*) – The number of sequences allowed in each transmission set

add_message_to_send (*message*)

Add a message to send. The message is converted to a sequenced message.

Parameters **message** (spinnman.messages.eieio.abstract_messages.AbstractEIEIOMessage) – The message to be added

is_empty()

Determine if there are no messages

Return type int

is_full

Determine if the number of messages sent is at the limit for the sequencing system

Return type bool

messages

The messages that have been added to the set

Return type iterable(`spinnman.messages.eieio.command_messages.
host_send_sequenced_data.HostSendSequencedData`)

send_stop_message()

Send a message to indicate the end of all the messages

update_last_received_sequence_number (*last_received_sequence_no*)

Updates the last received sequence number. If the sequence number is within the valid window, packets before the sequence number within the window are removed, and the last received sequence number is updated, thus moving the window for the next call. If the sequence number is not within the valid window, it is assumed to be invalid and so is ignored.

Parameters `last_received_sequence_no` (*int*) – The new sequence number

Returns True if update went ahead, False if it was ignored

Return type bool

class `spinn_front_end_common.interface.buffer_management.storage_objects.ChannelBufferState`

Bases: object

Stores information related to a single channel output buffering state, as it is retrieved at the end of a simulation on the SpiNNaker system.

Parameters

- **start_address** – start buffering area memory address (32 bits)
- **current_write** – address where data was last written (32 bits)
- **current_read** – address where data was last read (32 bits)
- **end_address** – The address of first byte after the buffer (32 bits)
- **region_id** – The ID of the region (8 bits)
- **missing_info** – True if the region overflowed during the simulation (8 bits)
- **last_buffer_operation** – Last operation performed on the buffer - read or write (8 bits)

ChannelBufferStateSize = 24

static `create_from_bytearray` (*data*)

`current_read`

`current_write`

```
end_address
is_state_updated
last_buffer_operation
missing_info
region_id
set_update_completed()
static size_of_channel_state()
start_address
update_last_operation(operation)
update_read_pointer(read_ptr)
```

```
class spinn_front_end_common.interface.buffer_management.storage_objects.EndBufferingState
```

Bases: object

Stores the buffering state at the end of a simulation.

Parameters

- **buffering_out_fsm_state** – Final sequence number received
- **list_channel_buffer_state** – a list of channel state, where each channel is stored in a ChannelBufferState object

```
buffering_out_fsm_state
channel_buffer_state(i)
get_missing_info_for_region(region_id)
get_state_for_region(region_id)
static size_of_region(n_regions_to_record)
```

```
class spinn_front_end_common.interface.buffer_management.storage_objects.SqliteDatabase(data)
```

Bases: `spinn_front_end_common.interface.buffer_management.storage_objects.abstract_database.AbstractDatabase`

Specific implementation of the Database for SQLite

Parameters **database_file** (*str*) – The name of a file that contains (or will contain) an SQLite database holding the data.

```
clear()
```

Clears the data for all regions.

Warning: This method will be removed when the database moves to keeping data after reset.

Return type None

```
close()
```

Signals that the database can be closed and will not be reused.

Once this is called any other method in this API is allowed to raise any kind of exception.

```
get_region_data(x, y, p, region)
```

Get the data stored for a given region of a given core

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data
- **x** – x coordinate of the chip
- **y** – y coordinate of the chip
- **p** – Core within the specified chip
- **region** – Region containing the data

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool) Get the data stored for a given region of a given core

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool)

store_data_in_region_buffer (*x, y, p, region, data*)

Store some information in the correspondent buffer class for a specific chip, core and region

Parameters

- **x** (*int*) – x coordinate of the chip
- **y** (*int*) – y coordinate of the chip
- **p** (*int*) – Core within the specified chip
- **region** (*int*) – Region containing the data to be stored
- **data** (*bytearray*) – data to be stored
- **x** – x coordinate of the chip
- **y** – y coordinate of the chip
- **p** – Core within the specified chip
- **region** – Region containing the data to be stored
- **data** – data to be stored

Submodules

spinn_front_end_common.interface.buffer_management.buffer_manager module

class `spinn_front_end_common.interface.buffer_management.buffer_manager.BufferManager` (*placements, tags, transceiver, extra_machine_managed, extra_machine_managed_chinese, fixed_report_folder, uses_a_report_folder, java_caller*)

Bases: `object`

Manager of send buffers.

Parameters

- **placements** (`Placements`) – The placements of the vertices
- **tags** (`Tags`) – The tags assigned to the vertices
- **transceiver** (`Transceiver`) – The transceiver to use for sending and receiving information
- **packet_gather_cores_to_ethernet_connection_map** – mapping of cores to
- **report_folder** (`str`) – The directory for reports which includes the file to use as an SQL database.
- **java_caller** (`JavaCaller`) – Support class to call Java or None to use python

add_receiving_vertex (`vertex`)

Add a vertex into the managed list for vertices which require buffers to be received from them during runtime.

Parameters **vertex** – the vertex to be managed

add_sender_vertex (`vertex`)

Add a vertex into the managed list for vertices which require buffers to be sent to them during runtime.

Parameters **vertex** (`AbstractSendsBuffersFromHost`) – the vertex to be managed

clear_recorded_data (`x, y, p, recording_region_id`)

Removes the recorded data stored in memory.

Parameters

- **x** (`int`) – placement x coordinate
- **y** (`int`) – placement y coordinate
- **p** (`int`) – placement p coordinate
- **recording_region_id** (`int`) – the recording region ID

get_data_by_placement (`placement, recording_region_id`)

Get the data container for all the data retrieved during the simulation from a specific region area of a core.

Parameters

- **placement** (`Placement`) – the placement to get the data from
- **recording_region_id** (`int`) – desired recording data region

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (`bytearray`, `bool`)

get_data_for_placements (*placements*, *progress=None*)

get_data_for_vertex (*placement*, *recording_region_id*)

It is no longer possible to get access to the data pointer.

Use `get_data_by_vertex` instead which returns the data that `pointer.read_all()` used to return the missing flag as before

load_initial_buffers ()

Load the initial buffers for the senders using memory writes.

reset ()

Resets the buffered regions to start transmitting from the beginning of its expected regions and clears the buffered out data files.

resume ()

Resets any data structures needed before starting running again.

sender_vertices

The vertices which are buffered.

stop ()

Indicates that the simulation has finished, so no further outstanding requests need to be processed.

spinn_front_end_common.interface.buffer_management.recording_utilities module

`spinn_front_end_common.interface.buffer_management.recording_utilities.get_last_sequence_number`

Read the last sequence number from the data

Parameters

- **placement** – The placement from which to read the sequence number
- **transceiver** – The transceiver to use to read the sequence number
- **recording_data_address** – The address of the recording data from which to read the number

Return type `int`

`spinn_front_end_common.interface.buffer_management.recording_utilities.get_recording_data_size`

Get the size of the recorded data to be reserved that doesn't

Parameters **n_recorded_regions** – The number of regions to be recorded

Return type `int`

`spinn_front_end_common.interface.buffer_management.recording_utilities.get_recording_header`

Get data to be written for the recording header

Parameters

- **recorded_region_sizes** – A list of sizes of each region to be recorded. A size of 0 is acceptable.
- **time_between_triggers** – The minimum time between requesting reads of any region
- **buffer_size_before_request** – The amount of buffer to fill before a read request is sent
- **ip_tags** – A list of IP tags to extract the buffer tag from
- **buffering_tag** – The tag to use for buffering requests

Returns An array of values to be written as the header

Return type list(int)

`spinn_front_end_common.interface.buffer_management.recording_utilities.get_recording_header`

Get the size of the data to be written for the recording header

Parameters **n_recorded_regions** – The number of regions to be recorded

`spinn_front_end_common.interface.buffer_management.recording_utilities.get_region_pointer` (`/`

Get a pointer to a recording region

Parameters

- **placement** – The placement from which to read the pointer
- **transceiver** – The transceiver to use to read the pointer
- **recording_data_address** – The address of the recording data from which to read the pointer
- **region** – The index of the region to get the pointer of

Return type int

Module contents

class `spinn_front_end_common.interface.buffer_management.BufferManager` (*placements, tags, transceiver, extra_monitor_cores, packet_gather_cores_to_ethernet_connection_map, extra_monitor_to_chip_mapping, machine, fixed_routes, uses_advanced_monitors, report_folder, java_caller=None*)

Bases: `object`

Manager of send buffers.

Parameters

- **placements** (*Placements*) – The placements of the vertices
- **tags** (*Tags*) – The tags assigned to the vertices
- **transceiver** (*Transceiver*) – The transceiver to use for sending and receiving information
- **packet_gather_cores_to_ethernet_connection_map** – mapping of cores to
- **report_folder** (*str*) – The directory for reports which includes the file to use as an SQL database.
- **java_caller** (*JavaCaller*) – Support class to call Java or None to use python

add_receiving_vertex (*vertex*)

Add a vertex into the managed list for vertices which require buffers to be received from them during runtime.

Parameters **vertex** – the vertex to be managed

add_sender_vertex (*vertex*)

Add a vertex into the managed list for vertices which require buffers to be sent to them during runtime.

Parameters **vertex** (*AbstractSendsBuffersFromHost*) – the vertex to be managed

clear_recorded_data (*x, y, p, recording_region_id*)

Removes the recorded data stored in memory.

Parameters

- **x** (*int*) – placement x coordinate
- **y** (*int*) – placement y coordinate
- **p** (*int*) – placement p coordinate
- **recording_region_id** (*int*) – the recording region ID

get_data_by_placement (*placement, recording_region_id*)

Get the data container for all the data retrieved during the simulation from a specific region area of a core.

Parameters

- **placement** (`Placement`) – the placement to get the data from
- **recording_region_id** (`int`) – desired recording data region

Returns an array contained all the data received during the simulation, and a flag indicating if any data was missing

Return type (bytearray, bool)

get_data_for_placements (*placements, progress=None*)

get_data_for_vertex (*placement, recording_region_id*)

It is no longer possible to get access to the data pointer.

Use `get_data_by_vertex` instead which returns the data that `pointer.read_all()` used to return the missing flag as before

load_initial_buffers ()

Load the initial buffers for the senders using memory writes.

reset ()

Resets the buffered regions to start transmitting from the beginning of its expected regions and clears the buffered out data files.

resume ()

Resets any data structures needed before starting running again.

sender_vertices

The vertices which are buffered.

stop ()

Indicates that the simulation has finished, so no further outstanding requests need to be processed.

spinn_front_end_common.interface.ds package

Submodules

`spinn_front_end_common.interface.ds.data_row_reader` module

`spinn_front_end_common.interface.ds.data_row_writer` module

`spinn_front_end_common.interface.ds.data_specification_targets` module

`spinn_front_end_common.interface.ds.ds_abstract_database` module

`spinn_front_end_common.interface.ds.ds_pretend_database` module

`spinn_front_end_common.interface.ds.ds_sqlite_database` module

`spinn_front_end_common.interface.ds.ds_write_info` module

Module contents

`spinn_front_end_common.interface.interface_functions` package

Submodules

`spinn_front_end_common.interface.interface_functions.application_finisher` module

`spinn_front_end_common.interface.interface_functions.application_runner` module

`spinn_front_end_common.interface.interface_functions.buffer_extractor` module

`spinn_front_end_common.interface.interface_functions.buffer_manager_creator` module

`spinn_front_end_common.interface.interface_functions.chip_iobuf_clearer` module

`spinn_front_end_common.interface.interface_functions.chip_iobuf_extractor` module

`spinn_front_end_common.interface.interface_functions.chip_provenance_updater` module

`spinn_front_end_common.interface.interface_functions.chip_runtime_updater` module

`spinn_front_end_common.interface.interface_functions.data_in_multicast_routing_generator`
module

`spinn_front_end_common.interface.interface_functions.database_interface` module

`spinn_front_end_common.interface.interface_functions.dsg_region_reloader` module

`spinn_front_end_common.interface.interface_functions.edge_to_n_keys_mapper` module

`spinn_front_end_common.interface.interface_functions.graph_binary_gatherer` module

`spinn_front_end_common.interface.interface_functions.graph_data_specification_writer` module

`spinn_front_end_common.interface.interface_functions.graph_measurer` module

`spinn_front_end_common.interface.interface_functions.graph_provenance_gatherer` module

`spinn_front_end_common.interface.interface_functions.hbp_allocator` module

`spinn_front_end_common.interface.interface_functions.hbp_max_machine_generator` module

`spinn_front_end_common.interface.interface_functions.host_execute_data_specification` module

`spinn_front_end_common.interface.interface_functions.insert_chip_power_monitors_to_graphs`
module

`spinn_front_end_common.interface.interface_functions.insert_edges_to_extra_monitor_functionality` module

`spinn_front_end_common.interface.interface_functions.insert_edges_to_live_packet_gatherers` module

`spinn_front_end_common.interface.interface_functions.insert_extra_monitor_vertices_to_graphs` module

`spinn_front_end_common.interface.interface_functions.insert_live_packet_gatherers_to_graphs` module

`spinn_front_end_common.interface.interface_functions.load_executable_images` module

`spinn_front_end_common.interface.interface_functions.load_fixed_routes` module

`spinn_front_end_common.interface.interface_functions.locate_executable_start_type` module

`spinn_front_end_common.interface.interface_functions.machine_generator` module

`spinn_front_end_common.interface.interface_functions.notification_protocol` module

`spinn_front_end_common.interface.interface_functions.placements_provenance_gatherer` module

`spinn_front_end_common.interface.interface_functions.pre_allocate_resources_for_chip_power_monitor` module

`spinn_front_end_common.interface.interface_functions.pre_allocate_resources_for_live_packet_gatherers` module

`spinn_front_end_common.interface.interface_functions.preallocate_resources_for_extra_monitor_support` module

`spinn_front_end_common.interface.interface_functions.process_partition_constraints` module

`spinn_front_end_common.interface.interface_functions.profile_data_gatherer` module

`spinn_front_end_common.interface.interface_functions.provenance_json_writer` module

`spinn_front_end_common.interface.interface_functions.provenance_xml_writer` module

`spinn_front_end_common.interface.interface_functions.router_provenance_gatherer` module

`spinn_front_end_common.interface.interface_functions.routing_setup` module

`spinn_front_end_common.interface.interface_functions.routing_table_loader` module

`spinn_front_end_common.interface.interface_functions.spalloc_allocator` module

`spinn_front_end_common.interface.interface_functions.spalloc_max_machine_generator` module

`spinn_front_end_common.interface.interface_functions.tags_loader` module

`spinn_front_end_common.interface.interface_functions.tdma_agenda_builder` module

`spinn_front_end_common.interface.interface_functions.virtual_machine_generator` module

`spinn_front_end_common.interface.interface_functions.write_memory_io_data` module

Module contents

`spinn_front_end_common.interface.profiling` package

Submodules

`spinn_front_end_common.interface.profiling.abstract_has_profile_data` module

class `spinn_front_end_common.interface.profiling.abstract_has_profile_data.AbstractHasProfileData`

Bases: `object`

Indicates an object that can record a profile

get_profile_data (*transceiver, placement*)

Get the profile data recorded during simulation

Return type `spinn_front_end_common.interface.profiling.profile_data.ProfileData`

`spinn_front_end_common.interface.profiling.profile_data` module

class `spinn_front_end_common.interface.profiling.profile_data.ProfileData` (*tag_labels*)

Bases: `object`

A container for profile data

Parameters **tag_labels** (*list(str)*) – A list of labels indexed by tag ID

DURATION = 1

START_TIME = 0

add_data (*data*)

Add profiling data read from the profile section

Parameters **data** (*bytearray*) – Data read from the profile section on the machine

get_mean_ms (*tag*)

Get the mean time in milliseconds spent on operations with the given tag

Parameters **tag** (*str*) – The tag to get the mean time for

Return type `float`

get_mean_ms_per_ts (*tag, run_time_ms, machine_time_step_ms*)

Get the mean time in milliseconds spent on operations with the given tag per timestep

Parameters

- **tag** (*str*) – The tag to get the data for
- **machine_time_step_ms** (*int*) – The time step of the simulation in microseconds
- **run_time_ms** (*float*) – The run time of the simulation in milliseconds

Return type float

get_mean_n_calls_per_ts (*tag, run_time_ms, machine_time_step_ms*)

Get the mean number of times the given tag was recorded per timestep

Parameters

- **tag** (*str*) – The tag to get the data for
- **machine_time_step_ms** (*int*) – The time step of the simulation in microseconds
- **run_time_ms** (*float*) – The run time of the simulation in milliseconds

Return type float

get_n_calls (*tag*)

Get the number of times the given tag was recorded

Parameters **tag** (*str*) – The tag to get the number of calls of

Return type int

tags

The tags recorded as labels

Return type list(str)

spinn_front_end_common.interface.profiling.profile_utils module

`spinn_front_end_common.interface.profiling.profile_utils.get_profile_region_size` (*n_samples*)

Get the size of the region of the profile data.

Parameters **n_samples** – number of different samples to record

Returns the size in bytes used by the profile region

`spinn_front_end_common.interface.profiling.profile_utils.get_profiling_data` (*profile_region, tag_labels, txrx, placement*)

Utility function to get profile data from a profile region.

Parameters

- **profile_region** – DSG region to get profiling data out of SDRAM
- **tag_labels** – labels for the profiling data
- **txrx** – SpiNNMan transceiver
- **placement** – placement

Returns *ProfileData*

`spinn_front_end_common.interface.profiling.profile_utils.reserve_profile_region` (*spec*,
re-
gion,
n_samples)

Reserves the profile region for recording the profile data.

Parameters

- **spec** – the DSG specification writer
- **region** – region ID for the profile data
- **n_samples** – number of elements being sampled

Return type None

`spinn_front_end_common.interface.profiling.profile_utils.write_profile_region_data` (*spec*,
re-
gion,
n_samples)

Writes the profile region data.

Parameters

- **spec** – the DSG specification writer
- **region** – region ID for the profile data
- **n_samples** – number of elements being sampled

Return type None

Module contents

class `spinn_front_end_common.interface.profiling.AbstractHasProfileData`

Bases: object

Indicates an object that can record a profile

get_profile_data (*transceiver*, *placement*)

Get the profile data recorded during simulation

Return type `spinn_front_end_common.interface.profiling.
profile_data.ProfileData`

class `spinn_front_end_common.interface.profiling.ProfileData` (*tag_labels*)

Bases: object

A container for profile data

Parameters **tag_labels** (*list* (*str*)) – A list of labels indexed by tag ID

DURATION = 1

START_TIME = 0

add_data (*data*)

Add profiling data read from the profile section

Parameters **data** (*bytearray*) – Data read from the profile section on the machine

get_mean_ms (*tag*)

Get the mean time in milliseconds spent on operations with the given tag

Parameters **tag** (*str*) – The tag to get the mean time for

Return type float

get_mean_ms_per_ts (*tag*, *run_time_ms*, *machine_time_step_ms*)

Get the mean time in milliseconds spent on operations with the given tag per timestep

Parameters

- **tag** (*str*) – The tag to get the data for
- **machine_time_step_ms** (*int*) – The time step of the simulation in microseconds
- **run_time_ms** (*float*) – The run time of the simulation in milliseconds

Return type float

get_mean_n_calls_per_ts (*tag*, *run_time_ms*, *machine_time_step_ms*)

Get the mean number of times the given tag was recorded per timestep

Parameters

- **tag** (*str*) – The tag to get the data for
- **machine_time_step_ms** (*int*) – The time step of the simulation in microseconds
- **run_time_ms** (*float*) – The run time of the simulation in milliseconds

Return type float

get_n_calls (*tag*)

Get the number of times the given tag was recorded

Parameters **tag** (*str*) – The tag to get the number of calls of

Return type int

tags

The tags recorded as labels

Return type list(str)

spinn_front_end_common.interface.provenance package

Submodules

spinn_front_end_common.interface.provenance.abstract_provides_local_provenance_data module

class spinn_front_end_common.interface.provenance.abstract_provides_local_provenance_data.

Bases: object

Indicates an object that provides locally obtained provenance data

get_local_provenance_data ()

Get an iterable of provenance data items

Returns iterable of ProvenanceDataItem

spinn_front_end_common.interface.provenance.abstract_provides_provenance_data_from_machine module**class** `spinn_front_end_common.interface.provenance.abstract_provides_provenance_data_from_machine`Bases: `object`

Indicates that an object provides provenance data retrieved from the machine

get_provenance_data_from_machine (*transceiver, placement*)

Get an iterable of provenance data items

Parameters

- **transceiver** – the SpinnMan interface object
- **placement** – the placement of the object

Returns iterable of `ProvenanceDataItem`**spinn_front_end_common.interface.provenance.pacman_provenance_extractor module****class** `spinn_front_end_common.interface.provenance.pacman_provenance_extractor.PacmanProvenanceExtractor`Bases: `object`Extracts Provenance data from a `PACMANAlgorithmExecutor`**clear** ()

Clears the provenance data store

Return type `None`**data_items**

Returns the provenance data items

Returns list of provenance data items.**Return type** `iterable(ProvenanceDataItem)`**extract_provenance** (*executor*)

Acquires the timings from PACMAN algorithms (provenance data)

Parameters **executor** – the PACMAN workflow executor**Return type** `None`**spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl module****class** `spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl`Bases: `spinn_front_end_common.interface.provenance.abstract_provides_provenance_data_from_machine.AbstractProvidesProvenanceDataFromMachine`

An implementation that gets provenance data from a region of ints on the machine.

NUM_PROVENANCE_DATA_ENTRIES = 5**class** **PROVENANCE_DATA_ENTRIES**Bases: `enum.Enum`**CALLBACK_QUEUE_OVERLOADED** = 1**DMA_QUEUE_OVERLOADED** = 2

```

MAX_NUMBER_OF_TIMER_TIC_OVERRUN = 4
TIMER_TIC_HAS_OVERRUN = 3
TRANSMISSION_EVENT_OVERFLOW = 0

get_provenance_data_from_machine (transceiver, placement)
    Get an iterable of provenance data items

    Parameters
        • transceiver – the SpinnMan interface object
        • placement – the placement of the object

    Returns iterable of ProvenanceDataItem

static get_provenance_data_size (n_additional_data_items)

reserve_provenance_data_region (spec)

```

Module contents

```

class spinn_front_end_common.interface.provenance.AbstractProvidesLocalProvenanceData
    Bases: object

    Indicates an object that provides locally obtained provenance data

    get_local_provenance_data ()
        Get an iterable of provenance data items

        Returns iterable of ProvenanceDataItem

class spinn_front_end_common.interface.provenance.AbstractProvidesProvenanceDataFromMachine
    Bases: object

    Indicates that an object provides provenance data retrieved from the machine

    get_provenance_data_from_machine (transceiver, placement)
        Get an iterable of provenance data items

        Parameters
            • transceiver – the SpinnMan interface object
            • placement – the placement of the object

        Returns iterable of ProvenanceDataItem

class spinn_front_end_common.interface.provenance.PacmanProvenanceExtractor
    Bases: object

    Extracts Provenance data from a PACMANAlgorithmExecutor

    clear ()
        Clears the provenance data store

        Return type None

    data_items
        Returns the provenance data items

        Returns list of provenance data items.

        Return type iterable(ProvenanceDataItem)

```

extract_provenance (*executor*)

Acquires the timings from PACMAN algorithms (provenance data)

Parameters **executor** – the PACMAN workflow executor

Return type None

class `spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachineImpl`
 Bases: `spinn_front_end_common.interface.provenance.abstract_provides_provenance_data_from_machine.AbstractProvidesProvenanceDataFromMachine`

An implementation that gets provenance data from a region of ints on the machine.

NUM_PROVENANCE_DATA_ENTRIES = 5

class **PROVENANCE_DATA_ENTRIES**

Bases: `enum.Enum`

CALLBACK_QUEUE_OVERLOADED = 1

DMA_QUEUE_OVERLOADED = 2

MAX_NUMBER_OF_TIMER_TIC_OVERRUN = 4

TIMER_TIC_HAS_OVERRUN = 3

TRANSMISSION_EVENT_OVERFLOW = 0

get_provenance_data_from_machine (*transceiver, placement*)

Get an iterable of provenance data items

Parameters

- **transceiver** – the SpinnMan interface object
- **placement** – the placement of the object

Returns iterable of `ProvenanceDataItem`

static **get_provenance_data_size** (*n_additional_data_items*)

reserve_provenance_data_region (*spec*)

spinn_front_end_common.interface.simulation package

Submodules

spinn_front_end_common.interface.simulation.simulation_utilities module

`spinn_front_end_common.interface.simulation.simulation_utilities.get_simulation_header_array`

Get data to be written to the simulation header

Parameters

- **binary_file_name** – The name of the binary of the application
- **machine_time_step** – The time step of the simulation
- **time_scale_factor** – The time scaling of the simulation

Returns An array of values to be written as the simulation header

Module contents

`spinn_front_end_common.interface.simulation.get_simulation_header_array` (*binary_file_name*,
machine_time_step,
time_scale_factor)

Get data to be written to the simulation header

Parameters

- **binary_file_name** – The name of the binary of the application
- **machine_time_step** – The time step of the simulation
- **time_scale_factor** – The time scaling of the simulation

Returns An array of values to be written as the simulation header

1.1.3.2 Submodules

1.1.3.3 `spinn_front_end_common.interface.abstract_spinnaker_base` module

1.1.3.4 `spinn_front_end_common.interface.config_handler` module

class `spinn_front_end_common.interface.config_handler.ConfigHandler` (*configfile*,
default_fault_config_paths,
validation_cfg)

Bases: `object`

Subclass of `AbstractSpinnaker` base that handles function only dependent of the config and the order its methods are called

child_folder (*parent*, *child_name*)

set_up_output_application_data_specifics (*n_calls_to_run*)

Parameters **n_calls_to_run** –

the counter of how many times run has been called. :type n_calls_to_run: int :return: the run folder for this simulation to hold app data

write_finished_file ()

1.1.3.5 `spinn_front_end_common.interface.java_caller` module

class `spinn_front_end_common.interface.java_caller.JavaCaller` (*json_folder*,
java_call,
java_spinnaker_path=None,
java_properties=None)

Bases: `object`

Support class that holds all the stuff for running stuff in Java.

This includes the work of preparing data for transmitting to Java and back.

This separates the choices of how to call the Java batch vs streaming, jar locations, parameters, etc from the rest of the python code.

Creates a java caller and checks the user/config parameters.

Parameters

- **json_folder** (*str*) – The location where the machine JSON is written.
- **java_call** (*str*) – Call to start java. Including the path if required.
- **java_spinnaker_path** – the path where the java code can be found. This must point to a local copy of <https://github.com/SpiNNakerManchester/JavaSpiNNaker>. It must also have been built! If None the assumption is that it is the same parent directory as <https://github.com/SpiNNakerManchester/SpiNNFrontEndCommon>.
- **java_properties** (*str*) – Optional properties that will be passed to Java. Must start with -D For example -Dlogging.level=DEBUG

:raise ConfigurationException if simple parameter checking fails.

execute_app_data_specification (*use_monitors*)

execute_data_specification ()

Writes all the data specs, uploading the result to the machine.

execute_system_data_specification ()

Writes all the data specs, uploading the result to the machine.

get_all_data ()

Gets all the data from the previously set placements and put these in the previously set database.

set_advanced_monitors (*placements, tags, monitor_cores, packet_gathers*)

Parameters

- **placements** (*pacman.model.placements.Placements*) – The placements of the vertices
- **tags** (*pacman.model.tags.Tags*) – The tags assigned to the vertices
- **monitor_cores** –
- **packet_gathers** –

Return type None

set_machine (*machine*)

Passes the machine in leaving this class to decide pass it to Java.

Parameters **machine** (*spinn_machine.machine.Machine*) – A machine Object

set_placements (*placements, transceiver*)

Passes in the placements leaving this class to decide pass it to Java.

This method may obtain extra information about the placements which is why it also needs the transceiver.

Currently the extra information extracted is recording region base address but this could change if recording region saved in the database.

Currently this method uses JSON but that may well change to using the database.

Parameters

- **placements** – The Placements Object
- **transceiver** – The Transceiver

set_report_folder (*report_folder*)

Passes the database file in.

Parameters **report_folder** (*str*) – Path to directory with SQLite databases and into which java will write

1.1.3.6 spinn_front_end_common.interface.simulator_state module

class spinn_front_end_common.interface.simulator_state.**Simulator_State** (*value*,
doc=")

Bases: enum.Enum

Different states the Simulator could be in.

FINISHED = 3

INIT = 0

IN_RUN = 1

RUN_FOREVER = 2

SHUTDOWN = 4

STOP_REQUESTED = 5

1.1.3.7 Module contents

1.1.4 spinn_front_end_common.mapping_algorithms package

1.1.4.1 Subpackages

spinn_front_end_common.mapping_algorithms.on_chip_router_table_compression package

Submodules

spinn_front_end_common.mapping_algorithms.on_chip_router_table_compression.mundy_on_chip_router_compression module

Module contents

1.1.4.2 Module contents

1.1.5 spinn_front_end_common.utilities package

1.1.5.1 Subpackages

spinn_front_end_common.utilities.connections package

Submodules

spinn_front_end_common.utilities.connections.live_event_connection module

class spinn_front_end_common.utilities.connections.live_event_connection.**LiveEventConnection**

Bases: `spinn_front_end_common.utilities.database.database_connection.DatabaseConnection`

A connection for receiving and sending live events from and to SpiNNaker

Parameters

- **live_packet_gather_label** – The label of the LivePacketGather vertex to which received events are being sent
- **receive_labels** (*iterable(str)*) – Labels of vertices from which live events will be received.
- **send_labels** (*iterable(str)*) – Labels of vertices to which live events will be sent
- **local_host** (*str*) – Optional specification of the local hostname or IP address of the interface to listen on
- **local_port** (*int*) – Optional specification of the local port to listen on. Must match the port that the toolchain will send the notification on (19999 by default)

add_init_callback (*label, init_callback*)

Add a callback to be called to initialise a vertex

Parameters

- **label** (*str*) – The label of the vertex to be notified about. Must be one of the vertices listed in the constructor
- **init_callback** (*function(str, int, float, float) -> None*) – A function to be called to initialise the vertex. This should take as parameters the label of the vertex, the number of neurons in the population, the run time of the simulation in milliseconds, and the simulation timestep in milliseconds

add_pause_stop_callback (*label, pause_stop_callback*)

Add a callback for the pause and stop state of the simulation

Parameters

- **label** (*str*) – the label of the function to be sent
- **pause_stop_callback** (*function(str, SpynnakerLiveEventConnection) -> None*) – A function to be called when the pause or stop message has been received. This function should take the label of the referenced vertex, and an instance of this class, which can be used to send events.

Return type None

add_receive_callback (*label, live_event_callback*)

Add a callback for the reception of live events from a vertex

Parameters

- **label** (*str*) – The label of the vertex to be notified about. Must be one of the vertices listed in the constructor
- **live_event_callback** (*function(str, int, list(int)) -> None*) – A function to be called when events are received. This should take as parameters the label of the vertex, the simulation timestep when the event occurred, and an array-like of atom IDs.

add_receive_label (*label*)

add_send_label (*label*)

add_start_callback (*label, start_callback*)

Add a callback for the start of the simulation

Parameters

- **start_callback** (*function(str, SpynnakerLiveEventConnection) -> None*) – A function to be called when the start message has been received. This function should take the label of the referenced vertex, and an instance of this class, which can be used to send events
- **label** (*str*) – the label of the function to be sent

add_start_resume_callback (*label, start_resume_callback*)

close ()

Closes the connection

Returns Nothing is returned

Return type None

Raises **None** – No known exceptions are raised

send_event (*label, atom_id, send_full_keys=False*)

Send an event from a single atom

Parameters

- **label** (*str*) – The label of the vertex from which the event will originate
- **atom_id** (*int*) – The ID of the atom sending the event
- **send_full_keys** (*bool*) – Determines whether to send full 32-bit keys, getting the key for each atom from the database, or whether to send 16-bit atom IDs directly

send_events (*label, atom_ids, send_full_keys=False*)

Send a number of events

Parameters

- **label** (*str*) – The label of the vertex from which the events will originate
- **atom_ids** (*list(int)*) – array-like of atom IDs sending events
- **send_full_keys** (*bool*) – Determines whether to send full 32-bit keys, getting the key for each atom from the database, or whether to send 16-bit atom IDs directly

Module contents

```
class spinn_front_end_common.utilities.connections.LiveEventConnection (live_packet_gather_label,
                                                                    re-
                                                                    ceive_labels=None,
                                                                    send_labels=None,
                                                                    lo-
                                                                    cal_host=None,
                                                                    lo-
                                                                    cal_port=19999,
                                                                    ma-
                                                                    chine_vertices=False)
```

Bases: `spinn_front_end_common.utilities.database.database_connection.DatabaseConnection`

A connection for receiving and sending live events from and to SpiNNaker

Parameters

- **live_packet_gather_label** – The label of the LivePacketGather vertex to which received events are being sent
- **receive_labels** (*iterable(str)*) – Labels of vertices from which live events will be received.
- **send_labels** (*iterable(str)*) – Labels of vertices to which live events will be sent
- **local_host** (*str*) – Optional specification of the local hostname or IP address of the interface to listen on
- **local_port** (*int*) – Optional specification of the local port to listen on. Must match the port that the toolchain will send the notification on (19999 by default)

add_init_callback (*label, init_callback*)

Add a callback to be called to initialise a vertex

Parameters

- **label** (*str*) – The label of the vertex to be notified about. Must be one of the vertices listed in the constructor
- **init_callback** (*function(str, int, float, float) -> None*) – A function to be called to initialise the vertex. This should take as parameters the label of the vertex, the number of neurons in the population, the run time of the simulation in milliseconds, and the simulation timestep in milliseconds

add_pause_stop_callback (*label, pause_stop_callback*)

Add a callback for the pause and stop state of the simulation

Parameters

- **label** (*str*) – the label of the function to be sent
- **pause_stop_callback** (*function(str, SpiNNakerLiveEventConnection) -> None*) – A function to be called when the pause or stop message has been received. This function should take the label of the referenced vertex, and an instance of this class, which can be used to send events.

Return type None

add_receive_callback (*label, live_event_callback*)

Add a callback for the reception of live events from a vertex

Parameters

- **label** (*str*) – The label of the vertex to be notified about. Must be one of the vertices listed in the constructor
- **live_event_callback** (*function(str, int, list(int)) -> None*) – A function to be called when events are received. This should take as parameters the label of the vertex, the simulation timestep when the event occurred, and an array-like of atom IDs.

add_receive_label (*label*)

add_send_label (*label*)

add_start_callback (*label, start_callback*)

Add a callback for the start of the simulation

Parameters

- **start_callback** (*function(str, SpynnakerLiveEventConnection) -> None*) – A function to be called when the start message has been received. This function should take the label of the referenced vertex, and an instance of this class, which can be used to send events
- **label** (*str*) – the label of the function to be sent

add_start_resume_callback (*label, start_resume_callback*)

close ()

Closes the connection

Returns Nothing is returned

Return type None

Raises **None** – No known exceptions are raised

send_event (*label, atom_id, send_full_keys=False*)

Send an event from a single atom

Parameters

- **label** (*str*) – The label of the vertex from which the event will originate
- **atom_id** (*int*) – The ID of the atom sending the event
- **send_full_keys** (*bool*) – Determines whether to send full 32-bit keys, getting the key for each atom from the database, or whether to send 16-bit atom IDs directly

send_events (*label, atom_ids, send_full_keys=False*)

Send a number of events

Parameters

- **label** (*str*) – The label of the vertex from which the events will originate
- **atom_ids** (*list(int)*) – array-like of atom IDs sending events
- **send_full_keys** (*bool*) – Determines whether to send full 32-bit keys, getting the key for each atom from the database, or whether to send 16-bit atom IDs directly

spinn_front_end_common.utilities.database package

Submodules

spinn_front_end_common.utilities.database.database_connection module

class spinn_front_end_common.utilities.database.database_connection.**DatabaseConnection** (*start_*
stop_
lo-
cal_p
lo-
cal_p

Bases: `spinnman.connections.udp_packet_connections.udp_connection.UDPConnection`

A connection from the toolchain which will be notified when the database has been written, and can then respond when the database has been read, and further wait for notification that the simulation has started.

Parameters

- **start_resume_callback_function** (*function()* -> *None*) – A function to be called when the start message has been received. This function should not take any parameters or return anything.
- **local_host** (*str*) – Optional specification of the local hostname or IP address of the interface to listen on
- **local_port** (*int*) – Optional specification of the local port to listen on. Must match the port that the toolchain will send the notification on (19999 by default)

add_database_callback (*database_callback_function*)

Add a database callback to be called when the database is ready.

Parameters **database_callback_function** (function(
`spinn_front_end_common.utilities.database.database_reader. DatabaseReader`) -> *None*) – A function to be called when the database message has been received. This function should take a single parameter, which will be a DatabaseReader object. Once the function returns, it will be assumed that the database has been read, and the return response will be sent.

Raises **SpinnmanIOException** – If anything goes wrong

close()

Closes the connection

Returns Nothing is returned

Return type *None*

Raises **None** – No known exceptions are raised

run()

spinn_front_end_common.utilities.database.database_reader module

class spinn_front_end_common.utilities.database.database_reader.**DatabaseReader** (*database_path*)
Bases: *object*

A reader for the database.

Parameters **database_path** (*str*) – The path to the database

close()

cursor

The database cursor. Allows custom SQL queries to be performed.

Return type `sqlite3.Cursor`

get_atom_id_to_key_mapping (*label*)

Get a mapping of atom ID to event key for a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns dictionary of event keys indexed by atom ID

Return type `dict(int, int)`

get_configuration_parameter_value (*parameter_name*)

Get the value of a configuration parameter

Parameters **parameter_name** (*str*) – The name of the parameter

Returns The value of the parameter

Return type `float`

get_ip_address (*x, y*)

Get an IP address to contact a chip

Parameters

- **x** – The x-coordinate of the chip
- **y** – The y-coordinate of the chip

Returns The IP address of the Ethernet to use to contact the chip

get_key_to_atom_id_mapping (*label*)

Get a mapping of event key to atom ID for a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns dictionary of atom IDs indexed by event key

Return type `dict(int, int)`

get_live_input_details (*label*)

Get the IP address and port where live input should be sent for a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port)

Return type `tuple(str, int)`

get_live_output_details (*label, receiver_label*)

Get the IP address, port and whether the SDP headers are to be stripped from the output from a vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port, strip SDP)

Return type `tuple(str, int, bool)`

get_machine_live_input_details (*label*)

Get the IP address and port where live input should be sent for a given machine vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port)

Return type tuple(str, int)

get_machine_live_input_key (*label*)

get_machine_live_output_details (*label*, *receiver_label*)

Get the IP address, port and whether the SDP headers are to be stripped from the output from a machine vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port, strip SDP)

Return type tuple(str, int, bool)

get_machine_live_output_key (*label*, *receiver_label*)

get_n_atoms (*label*)

Get the number of atoms in a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns The number of atoms

Return type int

get_placement (*label*)

Get the placement of a machine vertex with a given label

Parameters **label** (*str*) – The label of the vertex

Returns The x, y, p coordinates of the vertex

Return type tuple(int, int, int)

get_placements (*label*)

Get the placements of an application vertex with a given label

Parameters **label** – The label of the vertex

:type label:str :return: A list of x, y, p coordinates of the vertices :rtype: list(tuple(int, int, int))

spinn_front_end_common.utilities.database.database_writer module

class spinn_front_end_common.utilities.database.database_writer.**DatabaseWriter** (*database_directo*)

Bases: object

The interface for the database system for main front ends. Any special tables needed from a front end should be done by sub classes of this interface.

add_application_vertices (*application_graph*)

Parameters **application_graph** –

Return type None

add_machine_objects (*machine*)

Store the machine object into the database

Parameters **machine** – the machine object.

Return type None

add_placements (*placements*)

Adds the placements objects into the database

Parameters

- **placements** – the placements object
- **machine_graph** – the machine graph object

Return type None

add_routing_infos (*routing_infos, machine_graph*)

Adds the routing information (key masks etc) into the database

Parameters

- **routing_infos** – the routing information object
- **machine_graph** – the machine graph object

Return type None:

add_routing_tables (*routing_tables*)

Adds the routing tables into the database

Parameters **routing_tables** – the routing tables object

Return type None

add_system_params (*time_scale_factor, machine_time_step, runtime*)

Write system params into the database

Parameters

- **time_scale_factor** – the time scale factor used in timing
- **machine_time_step** – the machine time step used in timing
- **runtime** – the amount of time the application is to run for

add_tags (*machine_graph, tags*)

Adds the tags into the database

Parameters

- **machine_graph** – the machine graph object
- **tags** – the tags object

Return type None

add_vertices (*machine_graph, data_n_timesteps, graph_mapper, application_graph*)

Add the machine graph, graph mapper and application graph into the database.

Parameters

- **machine_graph** – the machine graph object
- **data_n_timesteps** – The number of timesteps for which data space will be reserved
- **graph_mapper** – the graph mapper object
- **application_graph** – the application graph object

Return type None

static auto_detect_database (*machine_graph*)

Auto detects if there is a need to activate the database system

Parameters **machine_graph** – the machine graph of the application problem space.

Returns a bool which represents if the database is needed

create_atom_to_event_id_mapping (*application_graph*, *machine_graph*, *routing_infos*, *graph_mapper*)

Parameters

- **application_graph** –
- **machine_graph** –
- **routing_infos** –
- **graph_mapper** –

Return type None

create_schema ()

database_path

Module contents

class `spinn_front_end_common.utilities.database.DatabaseConnection` (*start_resume_callback_function=None*, *stop_pause_callback_function=None*, *local_host=None*, *local_port=19999*)

Bases: `spinnman.connections.udp_packet_connections.udp_connection.UDPConnection`

A connection from the toolchain which will be notified when the database has been written, and can then respond when the database has been read, and further wait for notification that the simulation has started.

Parameters

- **start_resume_callback_function** (*function()* -> *None*) – A function to be called when the start message has been received. This function should not take any parameters or return anything.
- **local_host** (*str*) – Optional specification of the local hostname or IP address of the interface to listen on
- **local_port** (*int*) – Optional specification of the local port to listen on. Must match the port that the toolchain will send the notification on (19999 by default)

add_database_callback (*database_callback_function*)

Add a database callback to be called when the database is ready.

Parameters **database_callback_function** (*function*(*spinn_front_end_common.utilities.database.database_reader.DatabaseReader*) -> *None*) – A function to be called when the database message has been received. This function should take a single parameter, which will be a DatabaseReader object. Once the function returns, it will be assumed that the database has been read, and the return response will be sent.

Raises **SpinnmanIOException** – If anything goes wrong

close ()

Closes the connection

Returns Nothing is returned

Return type None

Raises **None** – No known exceptions are raised

run()

class `spinn_front_end_common.utilities.database.DatabaseReader(database_path)`
 Bases: `object`

A reader for the database.

Parameters **database_path** (*str*) – The path to the database

close()

cursor

The database cursor. Allows custom SQL queries to be performed.

Return type `sqlite3.Cursor`

get_atom_id_to_key_mapping (*label*)

Get a mapping of atom ID to event key for a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns dictionary of event keys indexed by atom ID

Return type `dict(int, int)`

get_configuration_parameter_value (*parameter_name*)

Get the value of a configuration parameter

Parameters **parameter_name** (*str*) – The name of the parameter

Returns The value of the parameter

Return type `float`

get_ip_address (*x, y*)

Get an IP address to contact a chip

Parameters

- **x** – The x-coordinate of the chip
- **y** – The y-coordinate of the chip

Returns The IP address of the Ethernet to use to contact the chip

get_key_to_atom_id_mapping (*label*)

Get a mapping of event key to atom ID for a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns dictionary of atom IDs indexed by event key

Return type `dict(int, int)`

get_live_input_details (*label*)

Get the IP address and port where live input should be sent for a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port)

Return type `tuple(str, int)`

get_live_output_details (*label, receiver_label*)

Get the IP address, port and whether the SDP headers are to be stripped from the output from a vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port, strip SDP)

Return type tuple(str, int, bool)

get_machine_live_input_details (*label*)

Get the IP address and port where live input should be sent for a given machine vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port)

Return type tuple(str, int)

get_machine_live_input_key (*label*)

get_machine_live_output_details (*label*, *receiver_label*)

Get the IP address, port and whether the SDP headers are to be stripped from the output from a machine vertex

Parameters **label** (*str*) – The label of the vertex

Returns tuple of (IP address, port, strip SDP)

Return type tuple(str, int, bool)

get_machine_live_output_key (*label*, *receiver_label*)

get_n_atoms (*label*)

Get the number of atoms in a given vertex

Parameters **label** (*str*) – The label of the vertex

Returns The number of atoms

Return type int

get_placement (*label*)

Get the placement of a machine vertex with a given label

Parameters **label** (*str*) – The label of the vertex

Returns The x, y, p coordinates of the vertex

Return type tuple(int, int, int)

get_placements (*label*)

Get the placements of an application vertex with a given label

Parameters **label** – The label of the vertex

:type label:str :return: A list of x, y, p coordinates of the vertices :rtype: list(tuple(int, int, int))

class spinn_front_end_common.utilities.database.DatabaseWriter (*database_directory*)

Bases: object

The interface for the database system for main front ends. Any special tables needed from a front end should be done by sub classes of this interface.

add_application_vertices (*application_graph*)

Parameters **application_graph** –

Return type None

add_machine_objects (*machine*)

Store the machine object into the database

Parameters **machine** – the machine object.

Return type None

add_placements (*placements*)

Adds the placements objects into the database

Parameters

- **placements** – the placements object
- **machine_graph** – the machine graph object

Return type None

add_routing_infos (*routing_infos, machine_graph*)

Adds the routing information (key masks etc) into the database

Parameters

- **routing_infos** – the routing information object
- **machine_graph** – the machine graph object

Return type None:

add_routing_tables (*routing_tables*)

Adds the routing tables into the database

Parameters **routing_tables** – the routing tables object

Return type None

add_system_params (*time_scale_factor, machine_time_step, runtime*)

Write system params into the database

Parameters

- **time_scale_factor** – the time scale factor used in timing
- **machine_time_step** – the machine time step used in timing
- **runtime** – the amount of time the application is to run for

add_tags (*machine_graph, tags*)

Adds the tags into the database

Parameters

- **machine_graph** – the machine graph object
- **tags** – the tags object

Return type None

add_vertices (*machine_graph, data_n_timesteps, graph_mapper, application_graph*)

Add the machine graph, graph mapper and application graph into the database.

Parameters

- **machine_graph** – the machine graph object
- **data_n_timesteps** – The number of timesteps for which data space will be reserved
- **graph_mapper** – the graph mapper object
- **application_graph** – the application graph object

Return type None

static auto_detect_database (*machine_graph*)

Auto detects if there is a need to activate the database system

Parameters *machine_graph* – the machine graph of the application problem space.

Returns a bool which represents if the database is needed

create_atom_to_event_id_mapping (*application_graph*, *machine_graph*, *routing_infos*,
graph_mapper)

Parameters

- *application_graph* –
- *machine_graph* –
- *routing_infos* –
- *graph_mapper* –

Return type None

create_schema ()

database_path

spinn_front_end_common.utilities.notification_protocol package

Submodules

spinn_front_end_common.utilities.notification_protocol.notification_protocol module

class spinn_front_end_common.utilities.notification_protocol.notification_protocol.**NotificationProtocol**

Bases: object

The protocol which hand shakes with external devices about the database and starting execution

close ()

Closes the thread pool

send_read_notification (*database_path*)

Sends notifications to all devices which have expressed an interest in when the database has been written

Parameters *database_path* – the path to the database file

Return type None

send_start_resume_notification ()

Either waits till all sources have confirmed read the database and are configured, and/or just sends the start notification (when the system is executing)

Return type None

send_stop_pause_notification ()

Sends the pause / stop notifications when the script has either finished or paused

Return type None

wait_for_confirmation ()

If asked to wait for confirmation, waits for all external systems to confirm that they are configured and have read the database

Return type None

spinn_front_end_common.utilities.notification_protocol.socket_address module

class spinn_front_end_common.utilities.notification_protocol.socket_address.**SocketAddress** (*n*

Bases: object

Data holder for a socket interface for notification protocol.

listen_port

The port to listen to for responses

notify_host_name

The notify host name

notify_port_no

The notify port no

Module contents

class spinn_front_end_common.utilities.notification_protocol.**NotificationProtocol** (*socket_address, wait_for_read*

Bases: object

The protocol which hand shakes with external devices about the database and starting execution

close()

Closes the thread pool

send_read_notification (*database_path*)

Sends notifications to all devices which have expressed an interest in when the database has been written

Parameters **database_path** – the path to the database file

Return type None

send_start_resume_notification()

Either waits till all sources have confirmed read the database and are configured, and/or just sends the start notification (when the system is executing)

Return type None

send_stop_pause_notification()

Sends the pause / stop notifications when the script has either finished or paused

Return type None

wait_for_confirmation()

If asked to wait for confirmation, waits for all external systems to confirm that they are configured and have read the database

Return type None

class spinn_front_end_common.utilities.notification_protocol.**SocketAddress** (*notify_host_name, notify_port_no, listen_port*)

Bases: object

Data holder for a socket interface for notification protocol.

listen_port

The port to listen to for responses

notify_host_name

The notify host name

notify_port_no

The notify port no

spinn_front_end_common.utilities.report_functions package

Submodules

spinn_front_end_common.utilities.report_functions.board_chip_report module

class spinn_front_end_common.utilities.report_functions.board_chip_report.**BoardChipReport**

Bases: object

Report on memory usage

AREA_CODE_REPORT_NAME = 'board_chip_report.txt'

spinn_front_end_common.utilities.report_functions.energy_report module

class spinn_front_end_common.utilities.report_functions.energy_report.**EnergyReport**

Bases: object

Creates a report about the approximate total energy consumed by a SpiNNaker job execution.

ENERGY_DETAILED_FILENAME = 'Detailed_energy_report.rpt'

ENERGY_SUMMARY_FILENAME = 'energy_summary_report.rpt'

JOULES_PER_SPIKE = 8e-10

JOULES_TO_KILOWATT_HOURS = 3600000

MILLIWATTS_FOR_BOXED_48_CHIP_FRAME_IDLE_COST = 0.0045833333

MILLIWATTS_FOR_FRAME_IDLE_COST = 0.117

MILLIWATTS_PER_CHIP_ACTIVE_OVERHEAD = 0.0006399999999999999

MILLIWATTS_PER_FPGA = 0.000584635

MILLIWATTS_PER_FRAME_ACTIVE_COST = 0.154163558

MILLIWATTS_PER_IDLE_CHIP = 0.00036

MILLIWATTS_PER_UNBOXED_48_CHIP_FRAME_IDLE_COST = 0.01666667

N_MONITORS_ACTIVE_DURING_COMMS = 2

`spinn_front_end_common.utilities.report_functions.fixed_route_from_machine_report` module

class `spinn_front_end_common.utilities.report_functions.fixed_route_from_machine_report.FixRouteFromMachineReport`
Bases: `object`
Generate a report of the fixed routes from the machine.

`spinn_front_end_common.utilities.report_functions.memory_map_on_host_chip_report` module

class `spinn_front_end_common.utilities.report_functions.memory_map_on_host_chip_report.MemoryMapOnHostChipReport`
Bases: `object`
Report on memory usage.

`spinn_front_end_common.utilities.report_functions.memory_map_on_host_report` module

class `spinn_front_end_common.utilities.report_functions.memory_map_on_host_report.MemoryMapOnHostReport`
Bases: `object`
Report on memory usage.

`spinn_front_end_common.utilities.report_functions.routing_table_from_machine_report` module

class `spinn_front_end_common.utilities.report_functions.routing_table_from_machine_report.RoutingTableFromMachineReport`
Bases: `object`

Module contents

class `spinn_front_end_common.utilities.report_functions.EnergyReport`
Bases: `object`
Creates a report about the approximate total energy consumed by a SpiNNaker job execution.

`ENERGY_DETAILED_FILENAME = 'Detailed_energy_report.rpt'`
`ENERGY_SUMMARY_FILENAME = 'energy_summary_report.rpt'`
`JOULES_PER_SPIKE = 8e-10`
`JOULES_TO_KILOWATT_HOURS = 3600000`
`MILLIWATTS_FOR_BOXED_48_CHIP_FRAME_IDLE_COST = 0.0045833333`
`MILLIWATTS_FOR_FRAME_IDLE_COST = 0.117`
`MILLIWATTS_PER_CHIP_ACTIVE_OVERHEAD = 0.0006399999999999999`
`MILLIWATTS_PER_FPGA = 0.000584635`
`MILLIWATTS_PER_FRAME_ACTIVE_COST = 0.154163558`
`MILLIWATTS_PER_IDLE_CHIP = 0.00036`
`MILLIWATTS_PER_UNBOXED_48_CHIP_FRAME_IDLE_COST = 0.01666667`
`N_MONITORS_ACTIVE_DURING_COMMS = 2`

class `spinn_front_end_common.utilities.report_functions.FixedRouteFromMachineReport`
 Bases: `object`

Generate a report of the fixed routes from the machine.

class `spinn_front_end_common.utilities.report_functions.MemoryMapOnHostChipReport`
 Bases: `object`

Report on memory usage.

class `spinn_front_end_common.utilities.report_functions.MemoryMapOnHostReport`
 Bases: `object`

Report on memory usage.

class `spinn_front_end_common.utilities.report_functions.RoutingTableFromMachineReport`
 Bases: `object`

spinn_front_end_common.utilities.scp package

Submodules

spinn_front_end_common.utilities.scp.clear_iobuf_process module

class `spinn_front_end_common.utilities.scp.clear_iobuf_process.ClearIOBUFProcess` (*connection_se*)
 Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

clear_iobuf (*core_subsets, n_cores*)

receive_response (*response*)

spinn_front_end_common.utilities.scp.scp_clear_iobuf_request module

class `spinn_front_end_common.utilities.scp.scp_clear_iobuf_request.SCPClearIOBUFRequest` (*x,*
y,
p,
des
ti-
na-
tion
ex-
pec)
 Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

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

spinn_front_end_common.utilities.scp.scp_update_runtime_request module

class spinn_front_end_common.utilities.scp.scp_update_runtime_request.SCPUpdateRuntimeRequest

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

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

spinn_front_end_common.utilities.scp.update_runtime_process module

class spinn_front_end_common.utilities.scp.update_runtime_process.UpdateRuntimeProcess(*connection_selector*)

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

receive_response(response)

update_runtime(current_time, run_time, infinite_run, core_subsets, n_cores)

Module contents

class spinn_front_end_common.utilities.scp.ClearIOBUFProcess(*connection_selector*)

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

clear_iobuf(core_subsets, n_cores)

receive_response(response)

class spinn_front_end_common.utilities.scp.SCPClearIOBUFRequest(*x, y, p, destination_port, expect_response=True*)

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

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

```
class spinn_front_end_common.utilities.scp.SCPUpdateRuntimeRequest (x, y,
                                                                    p,    cur-
                                                                    rent_time,
                                                                    run_time,
                                                                    infi-
                                                                    nite_run,
                                                                    destina-
                                                                    tion_port,
                                                                    ex-
                                                                    pect_response=True)
```

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

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

```
class spinn_front_end_common.utilities.scp.UpdateRuntimeProcess (connection_selector)
```

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

receive_response(response)

update_runtime(current_time, run_time, infinite_run, core_subsets, n_cores)

spinn_front_end_common.utilities.utility_objs package

Subpackages

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages package

Submodules

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.clear_reinjection_queue_message module

```
class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.clear_reinj
```

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to set the dropped packet reinjected packet types

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17

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

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.get_reinjection_status_message
module

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.get_reinject`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to get the status of the dropped packet reinjection

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17

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

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.get_reinject`

Bases: `spinnman.messages.scp.abstract_messages.scp_response.AbstractSCPResponse`

An SCP response to a request for the dropped packet reinjection status

read_data_bytestring (*data, offset*)

See `spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_data_bytestring()`

reinjection_functionality_status

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.load_application_mc_routes_message
module

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.load_applic`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to write the application multicast routes into the router.

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17

`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

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.load_system_mc_routes_message` module

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.load_system`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to write the system multicast routes into the router

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17
- **command_code** – the command code used by the extra monitor vertex for setting system multicast routes.

`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

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.reinjector_scp_commands` module

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.reinjector_scp_commands`

Bases: `enum.Enum`

SCP Command codes for reinjection

CLEAR = 6

EXIT = 5

```
GET_STATUS = 3
RESET_COUNTERS = 4
SET_PACKET_TYPES = 2
SET_ROUTER_EMERGENCY_TIMEOUT = 1
SET_ROUTER_TIMEOUT = 0
```

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.reset_counters_message` module

```
class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.reset_counters_message
```

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to reset the statistics counters of the dropped packet reinjection

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17

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

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_reinjection_packet_types_message` module

```
class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_reinjection_packet_types_message
```

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to set the dropped packet reinjected packet types

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

- **p**(*int*) – The processor running the extra monitor vertex, between 0 and 17
- **point_to_point**(*bool*) – If point to point should be set
- **multicast**(*bool*) – If multicast should be set
- **nearest_neighbour**(*bool*) – If nearest neighbour should be set
- **fixed_route**(*bool*) – If fixed route should be set

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

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_router_emergency_timeout_mes
module

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_router_`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to set the router emergency timeout for dropped packet reinjection

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
- **p**(*int*) – The processor running the extra monitor vertex, between 0 and 17
- **timeout_mantissa**(*int*) – The mantissa of the timeout value, between 0 and 15
- **timeout_exponent**(*int*) – The exponent of the timeout value, between 0 and 15

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

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_router_timeout_message module

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_router_timeout_message

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to the extra monitor core to set the router timeout for dropped packet reinjection

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17
- **timeout_mantissa** (*int*) – The mantissa of the timeout value, between 0 and 15
- **timeout_exponent** (*int*) – The exponent of the timeout value, between 0 and 15

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

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.speedup_in_scp_commands module

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.speedup_in_scp_commands

Bases: `enum.Enum`

SCP Command codes for data speed up in

LOAD_APPLICATION_MC_ROUTES = 7

LOAD_SYSTEM_MC_ROUTES = 8

SAVE_APPLICATION_MC_ROUTES = 6

Module contents

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.GetReinjectStatus

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to get the status of the dropped packet reinjection

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17

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

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.GetReinject`

Bases: `spinnman.messages.scp.abstract_messages.scp_response.AbstractSCPResponse`

An SCP response to a request for the dropped packet reinjection status

read_data_bytestring(data, offset)

See `spinnman.messages.scp.abstract_scp_response.AbstractSCPResponse.read_data_bytestring()`

reinjection_functionality_status

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.ResetCounter`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to reset the statistics counters of the dropped packet reinjection

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17

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

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.SetReinject`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to set the dropped packet reinjected packet types

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17
- **point_to_point** (*bool*) – If point to point should be set
- **multicast** (*bool*) – If multicast should be set
- **nearest_neighbour** (*bool*) – If nearest neighbour should be set
- **fixed_route** (*bool*) – If fixed route should be set

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

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.SetRouterEm`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to set the router emergency timeout for dropped packet reinjection

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17
- **timeout_mantissa** (*int*) – The mantissa of the timeout value, between 0 and 15
- **timeout_exponent** (*int*) – The exponent of the timeout value, between 0 and 15

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

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.**SetRouterTimeout**

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to the extra monitor core to set the router timeout for dropped packet reinjection

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
- **p**(*int*) – The processor running the extra monitor vertex, between 0 and 17
- **timeout_mantissa**(*int*) – The mantissa of the timeout value, between 0 and 15
- **timeout_exponent**(*int*) – The exponent of the timeout value, between 0 and 15

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

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.**ClearReinjection**

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to set the dropped packet reinjected packet types

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
- **p**(*int*) – The processor running the extra monitor vertex, between 0 and 17

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

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.**LoadApplication**

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to write the application multicast routes into the router.

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17

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

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.LoadSystemM`

Bases: `spinnman.messages.scp.abstract_messages.scp_request.AbstractSCPRequest`

An SCP Request to write the system multicast routes into the router

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
- **p** (*int*) – The processor running the extra monitor vertex, between 0 and 17
- **command_code** – the command code used by the extra monitor vertex for setting system multicast routes.

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

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes package

Submodules

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.clear_queue_process module

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.clear_queue`

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

reset_counters (*core_subsets*)

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_application_mc_routes_process
module

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_application_mc_routes_process

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

load_application_mc_routes (*core_subsets*)

Parameters **core_subsets** – sets of cores to send command to

Return type None

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_system_mc_routes_process
module

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_system_mc_routes_process

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

load_system_mc_routes (*core_subsets*)

Parameters **core_subsets** – sets of cores to send command to

Return type None

spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.read_status_process
module

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.read_status_process

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

get_reinjection_status (*x, y, p*)

get_reinjection_status_for_core_subsets (*core_subsets*)

handle_reinjection_status_response (*response*)

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.reset_counters_process` module

```
class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.reset_counters_process
    Bases:
        spinnman.processes.abstract_multi_connection_process.
        AbstractMultiConnectionProcess

    reset_counters (core_subsets)
```

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_packet_types_process` module

```
class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_packet_types_process
    Bases:
        spinnman.processes.abstract_multi_connection_process.
        AbstractMultiConnectionProcess

    set_packet_types (core_subsets, point_to_point, multicast, nearest_neighbour, fixed_route)
        Set what types of packets should be reinjected.
```

Parameters

- **core_subsets** – sets of cores to send command to
- **point_to_point** (*bool*) – If point-to-point should be set
- **multicast** (*bool*) – If multicast should be set
- **nearest_neighbour** (*bool*) – If nearest neighbour should be set
- **fixed_route** (*bool*) – If fixed route should be set
- **command_code** – The SCP command code

Return type None

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_router_emergency_timeout_process` module

```
class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_router_emergency_timeout_process
    Bases:
        spinnman.processes.abstract_multi_connection_process.
        AbstractMultiConnectionProcess

    set_timeout (mantissa, exponent, core_subsets)
```

`spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_router_timeout_process` module

```
class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_router_timeout_process
    Bases:
        spinnman.processes.abstract_multi_connection_process.
        AbstractMultiConnectionProcess

    set_timeout (mantissa, exponent, core_subsets)
```

Module contents

```

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.ReadStatus
    Bases: spinnman.processes.abstract_multi_connection_process.
            AbstractMultiConnectionProcess

    get_reinjection_status(x, y, p)

    get_reinjection_status_for_core_subsets(core_subsets)

    handle_reinjection_status_response(response)

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.ResetCounters
    Bases: spinnman.processes.abstract_multi_connection_process.
            AbstractMultiConnectionProcess

    reset_counters(core_subsets)

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.SetPacketTypes
    Bases: spinnman.processes.abstract_multi_connection_process.
            AbstractMultiConnectionProcess

    set_packet_types(core_subsets, point_to_point, multicast, nearest_neighbour, fixed_route)
        Set what types of packets should be reinjected.

        Parameters
        • core_subsets – sets of cores to send command to
        • point_to_point (bool) – If point-to-point should be set
        • multicast (bool) – If multicast should be set
        • nearest_neighbour (bool) – If nearest neighbour should be set
        • fixed_route (bool) – If fixed route should be set
        • command_code – The SCP command code

        Return type None

class spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.SetRouterError
    Bases: spinnman.processes.abstract_multi_connection_process.
            AbstractMultiConnectionProcess

    set_timeout(mantissa, exponent, core_subsets)

class spinn_front_end_end_common.utilities.utility_objs.extra_monitor_scp_processes.SetRouterTimeout
    Bases: spinnman.processes.abstract_multi_connection_process.
            AbstractMultiConnectionProcess

    set_timeout(mantissa, exponent, core_subsets)

class spinn_front_end_end_common.utilities.utility_objs.extra_monitor_scp_processes.ClearQueue
    Bases: spinnman.processes.abstract_multi_connection_process.
            AbstractMultiConnectionProcess

    reset_counters(core_subsets)

```

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.LoadApplication`

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

load_application_mc_routes (*core_subsets*)

Parameters `core_subsets` – sets of cores to send command to

Return type None

class `spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.LoadSystem`

Bases: `spinnman.processes.abstract_multi_connection_process.AbstractMultiConnectionProcess`

load_system_mc_routes (*core_subsets*)

Parameters `core_subsets` – sets of cores to send command to

Return type None

Submodules

`spinn_front_end_common.utilities.utility_objs.data_written` module

class `spinn_front_end_common.utilities.utility_objs.data_written.DataWritten` (*start_address*, *memory_used*, *memory_written*)

Bases: `object`

Describes data written to SpiNNaker.

memory_used

memory_written

start_address

spinn_front_end_common.utilities.utility_objs.dpri_flags module

class spinn_front_end_common.utilities.utility_objs.dpri_flags.**DPRIFlags** (*value*, *doc=""*)

Bases: enum.Enum

SCP Dropped Packet Reinjection (DPRI) packet type flags

FIXED_ROUTE = 8

MULTICAST = 1

NEAREST_NEIGHBOUR = 4

POINT_TO_POINT = 2

spinn_front_end_common.utilities.utility_objs.executable_finder module

class spinn_front_end_common.utilities.utility_objs.executable_finder.**ExecutableFinder** (*binary*, *include_common_binaries_folder*)

Bases: spinn_utilities.executable_finder.ExecutableFinder

Manages a set of folders in which to search for binaries, and allows for binaries to be discovered within this path. This adds a default location to look to the base class.

Parameters

- **binary_search_paths** (*iterable of str*) – The initial set of folders to search for binaries.
- **include_common_binaries_folder** (*bool*) – If True (i.e. the default), the spinn_front_end_common.common_model_binaries folder is searched for binaries. If you are not using the common models, or the model binary names conflict with your own, this parameter can be used to avoid this issue. Note that the folder will be appended to the value of binary_search_paths, so the common binary search path will be looked in last.

spinn_front_end_common.utilities.utility_objs.executable_targets module

class spinn_front_end_common.utilities.utility_objs.executable_targets.**ExecutableTargets**

Bases: spinnman.model.executable_targets.ExecutableTargets

add_processor (*binary*, *chip_x*, *chip_y*, *chip_p*, *executable_type=None*)

Add a processor to the executable targets

Parameters

- **binary** – the binary path for executable
- **chip_x** – the coordinate on the machine in terms of x for the chip
- **chip_y** – the coordinate on the machine in terms of y for the chip
- **chip_p** – the processor ID to place this executable on

Returns

add_subsets (*binary*, *subsets*, *executable_type=None*)

Add core subsets to a binary

Parameters

- **binary** – the path to the binary needed to be executed
- **subsets** – the subset of cores that the binary needs to be loaded on

Returns

executable_types_in_binary_set ()

get the executable types in the set of binaries

Returns iterable of the executable types in this binary set.

get_binaries_of_executable_type (*executable_type*)

get the binaries of a given a executable type

Parameters **execute_type** – the executable type enum value

Returns iterable of binaries with that executable type

get_n_cores_for_executable_type (*executable_type*)

returns the number of cores that the executable type is using

Parameters **executable_type** – the executable type for locating n cores of

Returns the number of cores using this executable type

spinn_front_end_common.utilities.utility_objs.executable_type module

```
class spinn_front_end_common.utilities.utility_objs.executable_type.ExecutableType (value,  
start_state,  
end_state,  
sup-  
ports_auto,  
doc="")
```

Bases: `enum.Enum`

The different types of executable from the perspective of how they are started and controlled.

NO_APPLICATION = 3

RUNNING = 0

SYNC = 1

SYSTEM = 4

USES_SIMULATION_INTERFACE = 2

spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters module**class** spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.**LivePacke**

Bases: object

Parameter holder for LPGs so that they can be instantiated at a later date.

board_address

hostname

key_prefix

message_type

number_of_packets_sent_per_time_step

partition_id

payload_as_time_stamps

payload_prefix

payload_right_shift

port

prefix_type

right_shift

strip_sdp

tag

use_payload_prefix

use_prefix

spinn_front_end_common.utilities.utility_objs.provenance_data_item module

class spinn_front_end_common.utilities.utility_objs.provenance_data_item.**ProvenanceDataItem**

Bases: object

Container for provenance data

Parameters

- **names** – A list of strings representing the naming hierarchy of this item
- **value** – The value of the item
- **report** – True if the item should be reported to the user
- **message** – The message to send to the end user if report is True

message

The message to report to the end user, or None if no message

names

The hierarchy of names of this bit of provenance data

report

True if this provenance data entry needs reporting to the end user

value

The value of the item

spinn_front_end_common.utilities.utility_objs.reinjection_status module

class spinn_front_end_common.utilities.utility_objs.reinjection_status.**ReInjectionStatus** (*data*, *offset*, *is_reinjecting_fixed_route*, *is_reinjecting_multicast*, *is_reinjecting_nearest_neighbour*, *is_reinjecting_point_to_point*)

Bases: object

Represents a status information from dropped packet reinjection

Parameters

- **data** (*str*) – The data containing the information
- **offset** (*int*) – The offset in the data where the information starts

is_reinjecting_fixed_route

True if re-injection of fixed-route packets is enabled

is_reinjecting_multicast

True if re-injection of multicast packets is enabled

is_reinjecting_nearest_neighbour

True if re-injection of nearest neighbour packets is enabled

is_reinjecting_point_to_point

True if re-injection of point-to-point packets is enabled

n_dropped_packet_overflows

Of the n_dropped_packets received, how many were lost due to not having enough space in the queue of packets to reinject

n_dropped_packets

The number of packets dropped by the router and received by the re injection functionality (may not fit in the queue though)

n_link_dumps

The number of times that when a dropped packet was caused due to a link failing to take the packet.

Returns int

n_missed_dropped_packets

The number of times that when a dropped packet was read it was found that another one or more packets had also been dropped, but had been missed

n_processor_dumps

The number of times that when a dropped packet was caused due to a processor failing to take the packet.

Returns int

n_reinjected_packets

Of the n_dropped_packets received, how many packets were successfully re injected

router_emergency_timeout

The WAIT2 timeout value of the router in cycles

router_emergency_timeout_parameters

The WAIT2 timeout value of the router as mantissa and exponent

router_timeout

The WAIT1 timeout value of the router in cycles

router_timeout_parameters

The WAIT1 timeout value of the router as mantissa and exponent

Module contents

```
class spinn_front_end_common.utilities.utility_objs.DataWritten(start_address,
                                                                memory_used,
                                                                mem-
                                                                ory_written)
```

Bases: object

Describes data written to SpiNNaker.

memory_used

memory_written

start_address

```
class spinn_front_end_common.utilities.utility_objs.DPRIFlags(value, doc="")
```

Bases: enum.Enum

SCP Dropped Packet Reinjection (DPRI) packet type flags

FIXED_ROUTE = 8

MULTICAST = 1

NEAREST_NEIGHBOUR = 4

POINT_TO_POINT = 2

class spinn_front_end_common.utilities.utility_objs.**ExecutableFinder** (*binary_search_paths=None, include_common_binaries_folder*)

Bases: spinn_utilities.executable_finder.ExecutableFinder

Manages a set of folders in which to search for binaries, and allows for binaries to be discovered within this path. This adds a default location to look to the base class.

Parameters

- **binary_search_paths** (*iterable of str*) – The initial set of folders to search for binaries.
- **include_common_binaries_folder** (*bool*) – If True (i.e. the default), the spinn_front_end_common.common_model_binaries folder is searched for binaries. If you are not using the common models, or the model binary names conflict with your own, this parameter can be used to avoid this issue. Note that the folder will be appended to the value of binary_search_paths, so the common binary search path will be looked in last.

class spinn_front_end_common.utilities.utility_objs.**ExecutableType** (*value, start_state, end_state, sup-ports_auto_pause_and_resume, doc=""*)

Bases: enum.Enum

The different types of executable from the perspective of how they are started and controlled.

NO_APPLICATION = 3

RUNNING = 0

SYNC = 1

SYSTEM = 4

USES_SIMULATION_INTERFACE = 2

```
class spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters (port,  
host-  
name,  
tag,  
board_address,  
strip_sdp,  
use_prefix,  
key_prefix,  
pre-  
fix_type,  
mes-  
sage_type,  
right_shift,  
pay-  
load_as_time_sta  
use_payload_pre  
pay-  
load_prefix,  
pay-  
load_right_shift,  
num-  
ber_of_packets_s  
par-  
ti-  
tion_id)
```

Bases: object

Parameter holder for LPGs so that they can be instantiated at a later date.

board_address

hostname

key_prefix

message_type

number_of_packets_sent_per_time_step

partition_id

payload_as_time_stamps

payload_prefix

payload_right_shift

port

prefix_type

right_shift

strip_sdp

tag

use_payload_prefix

use_prefix

```
class spinn_front_end_common.utilities.utility_objs.ProvenanceDataItem(names,  
                                                                    value,  
                                                                    re-  
                                                                    port=False,  
                                                                    mes-  
                                                                    sage=None)
```

Bases: object

Container for provenance data

Parameters

- **names** – A list of strings representing the naming hierarchy of this item
- **value** – The value of the item
- **report** – True if the item should be reported to the user
- **message** – The message to send to the end user if report is True

message

The message to report to the end user, or None if no message

names

The hierarchy of names of this bit of provenance data

report

True if this provenance data entry needs reporting to the end user

value

The value of the item

```
class spinn_front_end_common.utilities.utility_objs.ReInjectionStatus(data,  
                                                                    off-  
                                                                    set)
```

Bases: object

Represents a status information from dropped packet reinjection

Parameters

- **data** (*str*) – The data containing the information
- **offset** (*int*) – The offset in the data where the information starts

is_reinjecting_fixed_route

True if re-injection of fixed-route packets is enabled

is_reinjecting_multicast

True if re-injection of multicast packets is enabled

is_reinjecting_nearest_neighbour

True if re-injection of nearest neighbour packets is enabled

is_reinjecting_point_to_point

True if re-injection of point-to-point packets is enabled

n_dropped_packet_overflows

Of the n_dropped_packets received, how many were lost due to not having enough space in the queue of packets to reinject

n_dropped_packets

The number of packets dropped by the router and received by the re injection functionality (may not fit in the queue though)

n_link_dumps

The number of times that when a dropped packet was caused due to a link failing to take the packet.

Returns int

n_missed_dropped_packets

The number of times that when a dropped packet was read it was found that another one or more packets had also been dropped, but had been missed

n_processor_dumps

The number of times that when a dropped packet was caused due to a processor failing to take the packet.

Returns int

n_reinjected_packets

Of the n_dropped_packets received, how many packets were successfully re injected

router_emergency_timeout

The WAIT2 timeout value of the router in cycles

router_emergency_timeout_parameters

The WAIT2 timeout value of the router as mantissa and exponent

router_timeout

The WAIT1 timeout value of the router in cycles

router_timeout_parameters

The WAIT1 timeout value of the router as mantissa and exponent

class spinn_front_end_common.utilities.utility_objs.**ExecutableTargets**

Bases: `spinnman.model.executable_targets.ExecutableTargets`

add_processor (*binary, chip_x, chip_y, chip_p, executable_type=None*)

Add a processor to the executable targets

Parameters

- **binary** – the binary path for executable
- **chip_x** – the coordinate on the machine in terms of x for the chip
- **chip_y** – the coordinate on the machine in terms of y for the chip
- **chip_p** – the processor ID to place this executable on

Returns

add_subsets (*binary, subsets, executable_type=None*)

Add core subsets to a binary

Parameters

- **binary** – the path to the binary needed to be executed
- **subsets** – the subset of cores that the binary needs to be loaded on

Returns

executable_types_in_binary_set ()

get the executable types in the set of binaries

Returns iterable of the executable types in this binary set.

get_binaries_of_executable_type (*executable_type*)

get the binaries of a given a executable type

Parameters **execute_type** – the executable type enum value

Returns iterable of binaries with that executable type

get_n_cores_for_executable_type (*executable_type*)
returns the number of cores that the executable type is using

Parameters **executable_type** – the executable type for locating n cores of

Returns the number of cores using this executable type

1.1.5.2 Submodules

1.1.5.3 spinn_front_end_common.utilities.constants module

class `spinn_front_end_common.utilities.constants.BUFFERING_OPERATIONS`

Bases: `enum.Enum`

BUFFER_READ = 0

BUFFER_WRITE = 1

class `spinn_front_end_common.utilities.constants.SDP_PORTS`

Bases: `enum.Enum`

EXTRA_MONITOR_CORE_DATA_IN_SPEED_UP = 6

EXTRA_MONITOR_CORE_DATA_SPEED_UP = 5

EXTRA_MONITOR_CORE_REINJECTION = 4

INPUT_BUFFERING_SDP_PORT = 1

OUTPUT_BUFFERING_SDP_PORT = 2

RUNNING_COMMAND_SDP_PORT = 3

`spinn_front_end_common.utilities.constants.SDP_RUNNING_MESSAGE_CODES`

alias of `spinn_front_end_common.utilities.constants.SDP_RUNNING_MESSAGE_ID_CODES`

1.1.5.4 spinn_front_end_common.utilities.exceptions module

exception `spinn_front_end_common.utilities.exceptions.BufferableRegionTooSmall`

Bases: `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Raised when the SDRAM space of the region for buffered packets is too small to contain any packet at all

exception `spinn_front_end_common.utilities.exceptions.BufferedRegionNotPresent`

Bases: `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Raised when trying to issue buffered packets for a region not managed

exception `spinn_front_end_common.utilities.exceptions.ConfigurationException`

Bases: `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Raised when the front end determines a input parameter is invalid

exception `spinn_front_end_common.utilities.exceptions.ExecutableFailedToStartException`

Bases: `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Raised when an executable has not entered the expected state during start up

exception `spinn_front_end_common.utilities.exceptions.ExecutableFailedToStopException`

Bases: `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Raised when an executable has not entered the expected state during execution

exception `spinn_front_end_common.utilities.exceptions.ExecutableNotFoundException`

Bases: `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Raised when a specified executable could not be found

exception `spinn_front_end_common.utilities.exceptions.RallocException`

Bases: `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Raised when there are not enough routing table entries

exception `spinn_front_end_common.utilities.exceptions.SpinnFrontEndException`

Bases: `exceptions.Exception`

Raised when the front end detects an error

1.1.5.5 `spinn_front_end_common.utilities.failed_state` module

class `spinn_front_end_common.utilities.failed_state.FailedState`

Bases: `spinn_front_end_common.utilities.simulator_interface.SimulatorInterface`

add_socket_address (*socket_address*)

buffer_manager

config

graph_mapper

has_ran

increment_none_labelled_vertex_count

machine

machine_time_step

no_machine_time_steps

none_labelled_vertex_count

placements

run (*run_time*)

stop ()

tags

time_scale_factor

transceiver

use_virtual_board

verify_not_running ()

1.1.5.6 `spinn_front_end_common.utilities.function_list` module

1.1.5.7 `spinn_front_end_common.utilities.globals_variables` module

`spinn_front_end_common.utilities.globals_variables.get_not_running_simulator()`

`spinn_front_end_common.utilities.globals_variables.get_simulator()`

```
spinn_front_end_common.utilities.globals_variables.has_simulator()
spinn_front_end_common.utilities.globals_variables.set_failed_state(new_failed_state)
spinn_front_end_common.utilities.globals_variables.set_simulator(new_simulator)
spinn_front_end_common.utilities.globals_variables.unset_simulator()
```

1.1.5.8 spinn_front_end_common.utilities.helpful_functions module

`spinn_front_end_common.utilities.helpful_functions.convert_string_into_chip_and_core_subset`
 Translate a string list of cores into a core subset

Parameters `cores` (*str* or *None*) – string representing down cores formatted as x,y,p[:x,y,p]*

`spinn_front_end_common.utilities.helpful_functions.convert_time_diff_to_total_milliseconds`
 Convert between a time diff and total milliseconds.

Returns total milliseconds

Return type float

`spinn_front_end_common.utilities.helpful_functions.convert_vertices_to_core_subset` (*vertices*,
placements)
 Converts vertices into core subsets.

Parameters

- **extra_monitor_cores_to_set** – the vertices to convert to core subsets
- **placements** – the placements object

Returns the CoreSubSets of the vertices

`spinn_front_end_common.utilities.helpful_functions.determine_flow_states` (*executable_types*,
no_sync_changes)

Get the start and end states for these executable types.

Parameters

- **executable_types** (dict(*ExecutableType* -> any)) – the execute types to locate start and end states from
- **no_sync_changes** (*int*) – the number of times sync signals been sent

Returns dict of executable type to states.

Return type 2-tuple

`spinn_front_end_common.utilities.helpful_functions.emergency_recover_state_from_failure` (*txrx*,
app,
ver,
tex,
plac,
men)

Used to get at least *some* information out of a core when something goes badly wrong. Not a replacement for what abstract spinnaker base does.

Parameters

- **txrx** – The transceiver.
- **app_id** – The ID of the application.

- **vertex** (*spinn_front_end_common.abstract_models.AbstractHasAssociatedBinary*) – The vertex to retrieve the IOBUF from if it is suspected as being dead
- **placement** – Where the vertex is located.

`spinn_front_end_common.utilities.helpful_functions.emergency_recover_states_from_failure` (*txrx,*

app_id,
executable,
txrx,
app_id)

Used to get at least *some* information out of a core when something goes badly wrong. Not a replacement for what abstract spinnaker base does.

Parameters

- **txrx** – The transceiver.
- **app_id** – The ID of the application.
- **executable_targets** – The what/where mapping

`spinn_front_end_common.utilities.helpful_functions.flood_fill_binary_to_spinnaker` (*executable_targets,*

binary,
txrx,
app_id)

flood fills a binary to spinnaker on a given app_id given the executable targets and binary.

Parameters

- **executable_targets** – the executable targets object
- **binary** – the binary to flood fill
- **txrx** (Tranceiver) – spinnman instance
- **app_id** – the app id to load it on

Returns the number of cores it was loaded onto

`spinn_front_end_common.utilities.helpful_functions.generate_unique_folder_name` (*folder,*

file_name,
extension)

Generate a unique file name with a given extension in a given folder

Parameters

- **folder** – where to put this unique file
- **filename** – the name of the first part of the file without extension
- **extension** – extension of the file

Returns file path with a unique addition

`spinn_front_end_common.utilities.helpful_functions.get_ethernet_chip` (*machine,*

board_address)

Locate the chip with the given board IP address

Parameters

- **machine** – the SpiNNaker machine

- **board_address** – the board address to locate the chip of.

Returns The chip that supports that board address

Raises *ConfigurationException* – when that board address has no chip associated with it

`spinn_front_end_common.utilities.helpful_functions.locate_extra_monitor_mc_receiver` (*machine,*
place-
ment_x,
place-
ment_y,
packet_g)

`spinn_front_end_common.utilities.helpful_functions.locate_memory_region_for_placement` (*placem*
re-
gion,
transc)

Get the address of a region for a placement

Parameters

- **region** (*int*) – the region to locate the base address of
- **placement** (*Placement*) – the placement object to get the region address of
- **transceiver** (*Transceiver*) – the python interface to the SpiNNaker machine

`spinn_front_end_common.utilities.helpful_functions.read_config` (*config,* *section,*
item)

Get the string value of a config item, returning None if the value is “None”

`spinn_front_end_common.utilities.helpful_functions.read_config_boolean` (*config,*
sec-
tion,
item)

Get the boolean value of a config item, returning None if the value is “None”

`spinn_front_end_common.utilities.helpful_functions.read_config_int` (*config,*
section,
item)

Get the integer value of a config item, returning None if the value is “None”

`spinn_front_end_common.utilities.helpful_functions.read_data` (*x,* *y,* *ad-*
dress, *length,*
data_format,
transceiver)

Reads and converts a single data item from memory

Parameters

- **x** – chip x
- **y** – chip y
- **address** – base address of the SDRAM chip to read
- **length** – length to read
- **data_format** – the format to read memory
- **transceiver** – the SpinnMan interface

`spinn_front_end_common.utilities.helpful_functions.sort_out_downed_chips_cores_links` (*downed,*
downed,
downed,

Translate the down cores and down chips string into a form that spinnman can understand

Parameters

- **downed_cores** (*str or None*) – string representing down cores formatted as `x,y,p[:x,y,p]*`
- **downed_chips** (*str or None*) – string representing down chips formatted as `x,y[:x,y]*`
- **downed_links** – string representing down links formatted as `x,y,link[:x,y,link]*`

Returns a tuple of (set of (x, y) of down chips, set of (x, y, p) of down cores, set of ((x, y), link ID) of down links)

Return type tuple(set(tuple(int, int)), set(tuple(int, int, int)), set(tuple(tuple(int, int), int)))

```
spinn_front_end_common.utilities.helpful_functions.write_address_to_user0(txrx,
                                                                           x,
                                                                           y,
                                                                           p,
                                                                           ad-
                                                                           dress)
```

Writes the given address into the user_0 register of the given core.

Parameters

- **txrx** – The transceiver.
- **x** – Chip coordinate.
- **y** – Chip coordinate.
- **p** – Core ID on chip.
- **address** – Value to write (32-bit integer)

1.1.5.9 spinn_front_end_common.utilities.math_constants module

random math constants

1.1.5.10 spinn_front_end_common.utilities.simulator_interface module

```
class spinn_front_end_common.utilities.simulator_interface.SimulatorInterface
    Bases: object
```

```
    add_socket_address (socket_address)
```

```
    buffer_manager
```

```
    config
```

```
    graph_mapper
```

```
    has_ran
```

```
    increment_none_labelled_vertex_count
```

```
    machine
```

```
    machine_time_step
```

```
    no_machine_time_steps
```

```
    none_labelled_vertex_count
```

`placements`
`run`
`stop()`
`tags`
`time_scale_factor`
`transceiver`
`use_virtual_board`
`verify_not_running()`

1.1.5.11 Module contents

class `spinn_front_end_common.utilities.FailedState`

Bases: `spinn_front_end_common.utilities.simulator_interface.SimulatorInterface`

`add_socket_address(socket_address)`
`buffer_manager`
`config`
`graph_mapper`
`has_ran`
`increment_none_labelled_vertex_count`
`machine`
`machine_time_step`
`no_machine_time_steps`
`none_labelled_vertex_count`
`placements`
`run(run_time)`
`stop()`
`tags`
`time_scale_factor`
`transceiver`
`use_virtual_board`
`verify_not_running()`

class `spinn_front_end_common.utilities.SimulatorInterface`

Bases: `object`

`add_socket_address(socket_address)`
`buffer_manager`
`config`
`graph_mapper`

```

has_ran
increment_none_labelled_vertex_count
machine
machine_time_step
no_machine_time_steps
none_labelled_vertex_count
placements
run
stop()
tags
time_scale_factor
transceiver
use_virtual_board
verify_not_running()

```

1.1.6 spinn_front_end_common.utility_models package

1.1.6.1 Submodules

1.1.6.2 spinn_front_end_common.utility_models.chip_power_monitor module

class `spinn_front_end_common.utility_models.chip_power_monitor.ChipPowerMonitor` (*label, constraints, n_samples_per_recording, sampling_frequency*)

Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`, `spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`, `spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`

Represents idle time recording code in a application graph.

Parameters

- **label** (*str*) – vertex label
- **constraints** – constraints for the vertex
- **n_samples_per_recording** (*int*) – how many samples to take before recording to SDRAM the total
- **sampling_frequency** – how many microseconds between sampling

create_machine_vertex (*vertex_slice, resources_required, label=None, constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover

- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (*args, **kwargs)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_resources_used_by_atoms (*args, **kwargs)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type int

1.1.6.3 spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex module

class spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex.**ChipPowerMon**

Bases: pacman.model.graphs.machine.machine_vertex.MachineVertex,
spinn_front_end_common.abstract_models.abstract_has_associated_binary.
AbstractHasAssociatedBinary, spinn_front_end_common.abstract_models.
abstract_generates_data_specification.AbstractGeneratesDataSpecification,
spinn_front_end_common.interface.buffer_management.buffer_models.
abstract_receive_buffers_to_host.AbstractReceiveBuffersToHost

Machine vertex for C code representing functionality to record idle times in a machine graph.

class **CHIP_POWER_MONITOR_REGIONS**

Bases: enum.Enum

CONFIG = 1

RECORDING = 2

SYSTEM = 0

SAMPLE_RECORDING_REGION = 0

static binary_file_name()
Return the string binary file name

Returns basestring

static binary_start_type()
The type of binary that implements this vertex

Returns starttype enum

generate_data_specification(*args, **kwargs)
Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name()
Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()
Get the start type of the binary to be run.

Return type *ExecutableType*

get_recorded_data(placement, buffer_manager)
Get data from SDRAM given placement and buffer manager

Parameters

- **placement** – the location on machine to get data from
- **buffer_manager** – the buffer manager that might have data

Returns results

Return type numpy array with 1 dimension

get_recorded_region_ids()
Get the recording region IDs that have been recorded using buffering

Returns The region numbers that have active recording

Return type iterable(int)

get_recording_region_base_address(txrx, placement)
Get the recording region base address

Parameters

- **txrx** – the SpiNNMan instance
- **placement** – the placement object of the core to find the address of

Returns the base address of the recording region

static get_resources (*time_step*, *time_scale_factor*, *n_samples_per_recording*, *sampling_frequency*)

Get the resources used by this vertex

Returns Resource container

n_samples_per_recording

resources_required

The resources required by the vertex

Return type ResourceContainer

sampling_frequency

1.1.6.4 spinn_front_end_common.utility_models.command_sender module

class spinn_front_end_common.utility_models.command_sender.CommandSender (*label*, *constraints*)

Bases: pacman.model.graphs.application.application_vertex.

ApplicationVertex, spinn_front_end_common.abstract_models.

abstract_generates_data_specification.AbstractGeneratesDataSpecification,

spinn_front_end_common.abstract_models.abstract_has_associated_binary.

AbstractHasAssociatedBinary, spinn_front_end_common.abstract_models.

abstract_provides_outgoing_partition_constraints.AbstractProvidesOutgoingPartitionCons

A utility for sending commands to a vertex (possibly an external device) at fixed times in the simulation

add_commands (*start_resume_commands*, *pause_stop_commands*, *timed_commands*, *vertex_to_send_to*)

create_machine_vertex (*vertex_slice*, *resources_required*, *label=None*, *constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str* or *None*) – human readable label for the machine vertex
- **constraints** (*iterable* (*AbstractConstraint*)) – Constraints to be passed on to the machine vertex

edges_and_partitions ()

generate_data_specification (*spec*, *placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_outgoing_partition_constraints(partition)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters **partition** – An edge that comes out of this vertex

Returns A list of constraints

Return type *list(AbstractConstraint)*

get_resources_used_by_atoms(vertex_slice)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type *int*

1.1.6.5 spinn_front_end_common.utility_models.command_sender_machine_vertex module

class `spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex`

Bases: `pacman.model.graphs.machine.machine_vertex.MachineVertex`,
`spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.ProvidesProvenanceDataFromMachineImpl`,
`spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`,
`spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`,
`spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints.AbstractProvidesOutgoingPartitionConstraints`

BINARY_FILE_NAME = 'command_sender_multicast_source.aplx'

class **DATA_REGIONS**

Bases: `enum.Enum`

COMMANDS_AT_START_RESUME = 2

COMMANDS_AT_STOP_PAUSE = 3

COMMANDS_WITH_ARBITRARY_TIMES = 1

PROVENANCE_REGION = 4

SYSTEM_REGION = 0

add_commands (*start_resume_commands*, *pause_stop_commands*, *timed_commands*, *vertex_to_send_to*)

Add commands to be sent down a given edge

Parameters

- **start_resume_commands** (iterable(*spinn_front_end_common.utility_models.multi_cast_command.MultiCastCommand*)) – The commands to send when the simulation starts or resumes from pause
- **pause_stop_commands** (iterable(*spinn_front_end_common.utility_models.multi_cast_command.MultiCastCommand*)) – the commands to send when the simulation stops or pauses after running
- **timed_commands** (iterable(*spinn_front_end_common.utility_models.multi_cast_command.MultiCastCommand*)) – The commands to send at specific times
- **vertex_to_send_to** – The vertex these commands are to be sent to

edges_and_partitions ()**generate_data_specification** (*args, **kwargs)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None**get_binary_file_name** ()

Get the binary name to be run for this vertex.

Return type str

Return a string representation of the models binary

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType***get_edges_and_partitions** (*pre_vertex*, *edge_type*)**static get_n_command_bytes** (*commands*)**get_outgoing_partition_constraints** (*partition*)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters **partition** – An edge that comes out of this vertex**Returns** A list of constraints**Return type** list(*AbstractConstraint*)**get_timed_commands_bytes** ()**resources_required**

The resources required by the vertex

Return type *ResourceContainer*

1.1.6.6 spinn_front_end_common.utility_models.data_speed_up_packet_gatherer module

class spinn_front_end_common.utility_models.data_speed_up_packet_gatherer.DataSpeedUpPacketGatherer

Bases: pacman.model.graphs.application.application_vertex.ApplicationVertex, spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification, spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary

create_machine_vertex (*vertex_slice*, *resources_required*, *label=None*, *constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (*spec*, *placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()

Get the start type of the binary to be run.

Return type ExecutableType

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type ResourceContainer

Raises `None` – this method does not raise any known exception

machine_vertex

n_atoms

The number of atoms in the vertex

Return type `int`

1.1.6.7 `spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex` module

```
class spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.D
```

Bases: `enum.Enum`

RECEIVE_FINISHED = 2005

RECEIVE_FIRST_MISSING_SEQ = 2003

RECEIVE_MISSING_SEQ_DATA = 2004

SEND_DATA_TO_LOCATION = 200

SEND_DONE = 2002

SEND_SEQ_DATA = 2000

```
class spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.D
```

Bases: `enum.Enum`

CLEAR = 2000

MISSING_SEQ = 1001

START_MISSING_SEQ = 1000

START_SENDING = 100

```
class spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.D
```

Bases: `pacman.model.graphs.machine.machine_vertex.MachineVertex`,
`spinn_front_end_common.abstract_models.abstract_generates_data_specification`.
`AbstractGeneratesDataSpecification`, `spinn_front_end_common.abstract_models`.
`abstract_has_associated_binary.AbstractHasAssociatedBinary`,
`spinn_front_end_common.interface.provenance.abstract_provides_local_provenance_data`.
`AbstractProvidesLocalProvenanceData`

BASE_KEY = 4294967289

BASE_MASK = 4294967291

END_FLAG_KEY = 4294967286

END_FLAG_KEY_OFFSET = 3

```

FIRST_DATA_KEY = 4294967287
FIRST_DATA_KEY_OFFSET = 2
IN_REPORT_NAME = 'speeds_gained_in_speed_up_process.rpt'
LAST_MESSAGE_FLAG_BIT_MASK = 2147483648
LONG_TIMEOUT = (14, 14)
MISSING_SEQ_NUMS_END_FLAG = 4294967295
NEW_SEQ_KEY = 4294967288
NEW_SEQ_KEY_OFFSET = 1
OUT_REPORT_NAME = 'routers_used_in_speed_up_process.rpt'
SEQUENCE_NUMBER_MASK = 2147483647
SHORT_TIMEOUT = (1, 1)
TEMP_TIMEOUT = (15, 4)
TIMEOUT_PER_RECEIVE_IN_SECONDS = 1
TIME_OUT_FOR_SENDING_IN_SECONDS = 0.01
TRAFFIC_TYPE = 2
TRANSMISSION_THROTTLE_TIME = 1e-06
ZERO_TIMEOUT = (0, 0)

```

calculate_max_seq_num()

Deduce the max sequence number expected to be received

Returns int of the biggest sequence num expected

generate_data_specification(*args, **kwargs)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_data(placement, memory_address, length_in_bytes, fixed_routes)

Gets data from a given core and memory address.

Parameters

- **placement** – placement object for where to get data from
- **memory_address** – the address in SDRAM to start reading from
- **length_in_bytes** – the length of data to read in bytes

- **fixed_routes** – the fixed routes, used in the report of which chips were used by the speed up process

Returns byte array of the data

get_local_provenance_data()

Get an iterable of provenance data items

Returns iterable of `ProvenanceDataItem`

static load_application_routing_tables (*transceiver, extra_monitor_cores, placements*)

Set all chips to have application table loaded in the router

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type None

static load_system_routing_tables (*transceiver, extra_monitor_cores, placements*)

Set all chips to have the system table loaded in the router

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type None

static locate_correct_write_data_function_for_chip_location (*uses_advanced_monitors, machine, x, y, transceiver, ex-*

tra_monitor_cores_to_ethernet_conne

supports other components figuring out which gather and function to call for writing data onto spinnaker

Parameters

- **uses_advanced_monitors** (*bool*) – Whether the system is using advanced monitors
- **machine** – the SpiNNMachine instance
- **x** – the chip x coordinate to write data to
- **y** – the chip y coordinate to write data to
- **extra_monitor_cores_to_ethernet_connection_map** – mapping between cores and connections
- **transceiver** – the SpiNNMan instance

Returns a write function of either a LPG or the spinnMan

Return type func

resources_required

The resources required by the vertex

Return type `ResourceContainer`

send_data_into_spinnaker (*x, y, base_address, data, n_bytes=None, offset=0, cpu=0, is_filename=False*)

sends a block of data into SpiNNaker to a given chip

Parameters

- **x** – chip x for data
- **y** – chip y for data
- **base_address** – the address in SDRAM to start writing memory
- **data** – the data to write
- **n_bytes** – how many bytes to read, or None if not set
- **offset** – where in the data to start from
- **is_filename** (*bool*) – whether data is actually a file.

Return type None

set_cores_for_data_streaming (*transceiver, extra_monitor_cores, placements*)

Helper method for setting the router timeouts to a state usable for data streaming

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type None

static static_resources_required ()

static streaming (*gatherers, transceiver, extra_monitor_cores, placements*)

Helper method for setting the router timeouts to a state usable for data streaming via a Python context manager (i.e., using the ‘with’ statement).

Parameters

- **gatherers** – All the gatherers that are to be set
- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type a context manager

unset_cores_for_data_streaming (*transceiver, extra_monitor_cores, placements*)

Helper method for setting the router timeouts to a state usable for data streaming

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type None

`spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.ceildiv`

How to divide two possibly-integer numbers and round up.

1.1.6.8 `spinn_front_end_common.utility_models.extra_monitor_support` module

class `spinn_front_end_common.utility_models.extra_monitor_support.ExtraMonitorSupport` (*constraints*)
 Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`,
`spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`,
`spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`

create_machine_vertex (*vertex_slice, resources_required, label=None, constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type `None`

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type `str`

get_binary_start_type ()

Get the start type of the binary to be run.

Return type `ExecutableType`

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type `ResourceContainer`

Raises `None` – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type int

1.1.6.9 spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex module

class spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.**ExtraMoni**

Bases: `pacman.model.graphs.machine.machine_vertex.MachineVertex`,
`spinn_front_end_common.abstract_models.abstract_has_associated_binary`.
`AbstractHasAssociatedBinary`, `spinn_front_end_common.abstract_models`.
`abstract_generates_data_specification.AbstractGeneratesDataSpecification`

Parameters

- **constraints** – constraints on this vertex
- **reinject_multicast** – if we reinject multicast packets; defaults to value of *enable_reinjection* setting in configuration file
- **reinject_point_to_point** – if we reinject point-to-point packets
- **reinject_nearest_neighbour** – if we reinject nearest-neighbour packets
- **reinject_fixed_route** – if we reinject fixed route packets

clear_reinjection_queue (*transceiver, placements, extra_monitor_cores_to_set*)
 Clears the queues for reinjection

generate_data_specification (**args, **kwargs*)
 Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()
 Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()
 Get the start type of the binary to be run.

Return type *ExecutableType*

get_reinjection_status (*placements, transceiver*)
 Get the reinjection status from this extra monitor vertex

Parameters

- **transceiver** – the spinnMan interface
- **placements** – the placements object

Returns the reinjection status for this vertex

get_reinjection_status_for_vertices (*placements, extra_monitor_cores_for_data, transceiver*)

Get the reinjection status from a set of extra monitor cores

Parameters

- **placements** – the placements object
- **extra_monitor_cores_for_data** – the extra monitor cores to get status from
- **transceiver** – the spinnMan interface

Return type None

load_application_mc_routes (*placements, extra_monitor_cores_for_data, transceiver*)

Get the extra monitor cores to load up the application-based multicast routes (used by data in).

Parameters

- **placements** (*Placements*) – the placements object
- **extra_monitor_cores_for_data** (*iterable(ExtraMonitorSupportMachineVertex)*) – the extra monitor cores to get status from
- **transceiver** (*Transceiver*) – the spinnMan interface

Return type None

load_system_mc_routes (*placements, extra_monitor_cores_for_data, transceiver*)

Get the extra monitor cores to load up the system-based multicast routes (used by data in).

Parameters

- **placements** (*Placements*) – the placements object
- **extra_monitor_cores_for_data** (*iterable(ExtraMonitorSupportMachineVertex)*) – the extra monitor cores to get status from
- **transceiver** (*Transceiver*) – the spinnMan interface

Return type None

placement

reinject_fixed_route

reinject_multicast

reinject_nearest_neighbour

reinject_point_to_point

reset_reinjection_counters (*transceiver, placements, extra_monitor_cores_to_set*)

Resets the counters for reinjection

resources_required

The resources required by the vertex

Return type *ResourceContainer*

set_reinjection_packets (*placements*, *extra_monitor_cores_for_data*, *transceiver*,
point_to_point=None, *multicast=None*, *nearest_neighbour=None*,
fixed_route=None)

Parameters

- **placements** – placements object
- **extra_monitor_cores_for_data** – the extra monitor cores to set the packets of
- **transceiver** – spinnman instance
- **point_to_point** (*bool* or *None*) – If point to point should be set, or None if left as before
- **multicast** (*bool* or *None*) – If multicast should be set, or None if left as before
- **nearest_neighbour** (*bool* or *None*) – If nearest neighbour should be set, or None if left as before
- **fixed_route** (*bool* or *None*) – If fixed route should be set, or None if left as before.

Return type None

set_router_emergency_timeout (*timeout*, *transceiver*, *placements*, *extra_monitor_cores_to_set*)

Sets the timeout of the routers

Parameters

- **timeout** ((*int*, *int*)) – The mantissa and exponent of the timeout value, each between 0 and 15
- **transceiver** (Transceiver) – the spinnMan instance
- **placements** (Placements) – the placements object
- **extra_monitor_cores_to_set** – the set of vertices to change the local chip for.

set_router_time_outs (*timeout*, *transceiver*, *placements*, *extra_monitor_cores_to_set*)

Supports setting of the router time outs for a set of chips via their extra monitor cores.

Parameters

- **timeout** ((*int*, *int*)) – The mantissa and exponent of the timeout value, each between 0 and 15
- **transceiver** (Transceiver) – the spinnman interface
- **placements** (Placements) – placements object
- **extra_monitor_cores_to_set** – which vertices to use

Return type None

static static_get_binary_file_name ()

static static_get_binary_start_type ()

static static_resources_required ()

1.1.6.10 spinn_front_end_common.utility_models.live_packet_gather module

```
class spinn_front_end_common.utility_models.live_packet_gather.LivePacketGather (hostname,  
                                         port,  
                                         board_address=  
                                         tag=None,  
                                         strip_sdp=True,  
                                         use_prefix=False,  
                                         key_prefix=None,  
                                         pre-  
                                         fix_type=None,  
                                         mes-  
                                         sage_type=<EnumMeta  
                                         2>,  
                                         right_shift=0,  
                                         pay-  
                                         load_as_time_s  
                                         use_payload_p  
                                         pay-  
                                         load_prefix=No  
                                         pay-  
                                         load_right_shif  
                                         num-  
                                         ber_of_packets,  
                                         con-  
                                         straints=None,  
                                         la-  
                                         bel=None)
```

Bases: `pacman.model.graphs.application.application_vertex.
ApplicationVertex, spinn_front_end_common.abstract_models.
abstract_generates_data_specification.AbstractGeneratesDataSpecification,
spinn_front_end_common.abstract_models.abstract_has_associated_binary.
AbstractHasAssociatedBinary`

A model which stores all the events it receives during a timer tick and then compresses them into Ethernet packets and sends them out of a SpiNNaker machine.

create_machine_vertex (*vertex_slice, resources_required, label=None, constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type int

1.1.6.11 spinn_front_end_common.utility_models.live_packet_gather_machine_vertex module

class spinn_front_end_common.utility_models.live_packet_gather_machine_vertex.**LivePacketGat**

Bases: `pacman.model.graphs.machine.machine_vertex.MachineVertex`,

```
spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl
ProvidesProvenanceDataFromMachineImpl, spinn_front_end_common.
abstract_models.abstract_generates_data_specification.
AbstractGeneratesDataSpecification, spinn_front_end_common.abstract_models.
abstract_has_associated_binary.AbstractHasAssociatedBinary,
spinn_front_end_common.abstract_models.abstract_supports_database_injection.
AbstractSupportsDatabaseInjection
```

N_ADDITIONAL_PROVENANCE_ITEMS = 2

TRAFFIC_IDENTIFIER = 'LPG_EVENT_STREAM'

generate_data_specification (*args, **kwargs)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType*

static get_cpu_usage ()

Get the CPU used by this vertex

Returns 0

Return type int

static get_dtcu_usage ()

Get the DTCU used by this vertex

get_provenance_data_from_machine (transceiver, placement)

Get provenance from the machine

Parameters

- **transceiver** – spinnman interface to the machine
- **placement** – the location of this vertex on the machine

static get_sdram_usage ()

Get the SDRAM used by this vertex

is_in_injection_mode

Whether this vertex is actually in injection mode.

resources_required

The resources required by the vertex

Return type *ResourceContainer*

1.1.6.12 spinn_front_end_common.utility_models.multi_cast_command module

```
class spinn_front_end_common.utility_models.multi_cast_command.MultiCastCommand(key,
                                                                              pay-
                                                                              load=None,
                                                                              time=None,
                                                                              re-
                                                                              peat=0,
                                                                              de-
                                                                              lay_between_repeats=0)
```

Bases: object

A command to be sent to a vertex.

Parameters

- **key** (*int*) – The key of the command
- **payload** (*int*) – The payload of the command
- **time** (*int*) – The time within the simulation at which to send the command, or None if this is not a timed command
- **repeat** (*int*) – The number of times that the command should be repeated after sending it once. This could be used to ensure that the command is sent despite lost packets. Must be between 0 and 65535
- **delay_between_repeats** (*int*) – The amount of time in microseconds to wait between sending repeats of the same command. Must be between 0 and 65535, and must be 0 if repeat is 0

Raises **SpynnakerException** – If the repeat or delay are out of range

delay_between_repeats

is_payload

Determine if this command has a payload. By default, this returns True if the payload passed in to the constructor is not None, but this can be overridden to indicate that a payload will be generated, despite None being passed to the constructor

Returns True if there is a payload, False otherwise

Return type bool

is_timed

key

payload

Get the payload of the command.

Returns The payload of the command, or None if there is no payload

Return type int

repeat

time

1.1.6.13 spinn_front_end_common.utility_models.reverse_ip_tag_multi_cast_source module

class `spinn_front_end_common.utility_models.reverse_ip_tag_multi_cast_source.ReverseIpTagM`

Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`, `spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`, `spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`, `spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints.AbstractProvidesOutgoingPartitionConstraints`, `spinn_front_end_common.abstract_models.impl.provides_key_to_atom_mapping_impl.ProvidesKeyToAtomMappingImpl`

A model which will allow events to be injected into a SpiNNaker machine and converted into multicast packets.

Parameters

- **n_keys** – The number of keys to be sent via this multicast source
- **label** – The label of this vertex
- **constraints** – Any initial constraints to this vertex
- **board_address** – The IP address of the board on which to place this vertex if receiving data, either buffered or live (by default, any board is chosen)
- **receive_port** – The port on the board that will listen for incoming event packets (default is to disable this feature; set a value to enable it)
- **receive_sdp_port** – The SDP port to listen on for incoming event packets (defaults to 1)
- **receive_tag** – The IP tag to use for receiving live events (uses any by default)
- **receive_rate** – The estimated rate of packets that will be sent by this source

- **virtual_key** – The base multicast key to send received events with (assigned automatically by default)
- **prefix** – The prefix to “or” with generated multicast keys (default is no prefix)
- **prefix_type** – Whether the prefix should apply to the upper or lower half of the multicast keys (default is upper half)
- **check_keys** – True if the keys of received events should be verified before sending (default False)
- **send_buffer_times** – An array of arrays of times at which keys should be sent (one array for each key, default disabled)
- **send_buffer_partition_id** – The ID of the partition containing the edges down which the events are to be sent
- **send_buffer_max_space** – The maximum amount of space to use of the SDRAM on the machine (default is 1MB)
- **send_buffer_space_before_notify** – The amount of space free in the sending buffer before the machine will ask the host for more data (default setting is optimised for most cases)
- **buffer_notification_ip_address** – The IP address of the host that will send new buffers (must be specified if a send buffer is specified or if recording will be used)
- **buffer_notification_port** – The port that the host that will send new buffers is listening on (must be specified if a send buffer is specified, or if recording will be used)
- **buffer_notification_tag** – The IP tag to use to notify the host about space in the buffer (default is to use any tag)

create_machine_vertex (*vertex_slice*, *resources_required*, *label=None*, *constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

enable_recording (*new_state=True*)

generate_data_specification (*spec*, *placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_outgoing_partition_constraints(partition)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters **partition** – An edge that comes out of this vertex

Returns A list of constraints

Return type list(*AbstractConstraint*)

get_resources_used_by_atoms(vertex_slice)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type int

send_buffer_times

1.1.6.14 spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex module

class spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex

Bases: *pacman.model.graphs.machine.machine_vertex.MachineVertex*, *spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification*, *spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary*, *spinn_front_end_common.abstract_models.abstract_supports_database_injection.AbstractSupportsDatabaseInjection*, *spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.ProvidesProvenanceDataFromMachineImpl*, *spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints.AbstractProvidesOutgoingPartitionConstraints*, *spinn_front_end_common.interface.buffer_management.buffer_models.sends_buffers_from_host_pre_buffered_impl.SendsBuffersFromHostPreBufferedImpl*, *spinn_front_end_common.interface.buffer_management.buffer_models.abstract_receive_buffers_to_host.AbstractReceiveBuffersToHost*, *spinn_front_end_common.abstract_models.abstract_recordable.AbstractRecordable*

A model which allows events to be injected into SpiNNaker and converted in to multicast packets

enable_recording(new_state=True)

Enable recording of the keys sent

generate_data_specification(*args, **kwargs)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None**get_binary_file_name** ()

Get the binary name to be run for this vertex.

Return type str**get_binary_start_type** ()

Get the start type of the binary to be run.

Return type *ExecutableType***static get_cpu_usage** ()**static get_dtcn_usage** ()**get_outgoing_partition_constraints** (*partition*)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters **partition** – An edge that comes out of this vertex**Returns** A list of constraints**Return type** list(*AbstractConstraint*)**get_provenance_data_from_machine** (*transceiver, placement*)

Get an iterable of provenance data items

Parameters

- **transceiver** – the SpinnMan interface object
- **placement** – the placement of the object

Returns iterable of *ProvenanceDataItem***get_recorded_region_ids** ()

Get the recording region IDs that have been recorded using buffering

Returns The region numbers that have active recording**Return type** iterable(int)**get_recording_region_base_address** (*txrx, placement*)

Get the recording region base address

Parameters

- **txrx** – the SpiNNMan instance
- **placement** – the placement object of the core to find the address of

Returns the base address of the recording region**get_region_buffer_size** (*region*)

Get the size of the buffer to be used in SDRAM on the machine for the region in bytes

Parameters **region** (*int*) – The region to get the buffer size of**Returns** The size of the buffer space in bytes**Return type** int

```
static get_sdram_usage (send_buffer_times, recording_enabled, machine_time_step, receive_rate, n_keys)

is_in_injection_mode
    Whether this vertex is actually in injection mode.

is_recording ()
    Deduce if the recorder is actually recording

mask

static max_send_buffer_keys_per_timestep (send_buffer_times, n_keys)

static n_regions_to_allocate (send_buffering, recording)
    Get the number of regions that will be allocated

static recording_sdram_per_timestep (machine_time_step, is_recording, receive_rate, send_buffer_times, n_keys)

resources_required
    The resources required by the vertex

    Return type ResourceContainer

static send_buffer_sdram_per_timestep (send_buffer_times, n_keys)

send_buffer_times

send_buffers

update_buffer (arg)

virtual_key
```

1.1.6.15 Module contents

```
class spinn_front_end_common.utility_models.CommandSender (label, constraints)
```

Bases: [pacman.model.graphs.application.application_vertex.ApplicationVertex](#), [spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification](#), [spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary](#), [spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints.AbstractProvidesOutgoingPartitionConstraints](#)

A utility for sending commands to a vertex (possibly an external device) at fixed times in the simulation

```
add_commands (start_resume_commands, pause_stop_commands, timed_commands, vertex_to_send_to)
```

```
create_machine_vertex (vertex_slice, resources_required, label=None, constraints=None)
```

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** ([Slice](#)) – The slice of atoms that the machine vertex will cover
- **resources_required** ([ResourceContainer](#)) – the resources used by the machine vertex
- **label** (*str* or *None*) – human readable label for the machine vertex
- **constraints** (*iterable* ([AbstractConstraint](#))) – Constraints to be passed on to the machine vertex

```
edges_and_partitions ()
```

generate_data_specification (*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_outgoing_partition_constraints (*partition*)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters **partition** – An edge that comes out of this vertex

Returns A list of constraints

Return type list(*AbstractConstraint*)

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type int

```
class spinn_front_end_common.utility_models.CommandSenderMachineVertex (label,  
                                                                    con-  
                                                                    straints)
```

```
Bases:      pacman.model.graphs.machine.machine_vertex.MachineVertex,  
spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.  
ProvidesProvenanceDataFromMachineImpl,      spinn_front_end_common.  
abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary,  
spinn_front_end_common.abstract_models.abstract_generates_data_specification.  
AbstractGeneratesDataSpecification,      spinn_front_end_common.  
abstract_models.abstract_provides_outgoing_partition_constraints.  
AbstractProvidesOutgoingPartitionConstraints
```

```
BINARY_FILE_NAME = 'command_sender_multicast_source.aplx'
```

```
class DATA_REGIONS
```

```
Bases: enum.Enum
```

```
COMMANDS_AT_START_RESUME = 2
```

`COMMANDS_AT_STOP_PAUSE = 3`

`COMMANDS_WITH_ARBITRARY_TIMES = 1`

`PROVENANCE_REGION = 4`

`SYSTEM_REGION = 0`

`add_commands (start_resume_commands, pause_stop_commands, timed_commands, vertex_to_send_to)`

Add commands to be sent down a given edge

Parameters

- **start_resume_commands** (iterable(`spinn_front_end_common.utility_models.multi_cast_command.MultiCastCommand`)) – The commands to send when the simulation starts or resumes from pause
- **pause_stop_commands** (iterable(`spinn_front_end_common.utility_models.multi_cast_command.MultiCastCommand`)) – the commands to send when the simulation stops or pauses after running
- **timed_commands** (iterable(`spinn_front_end_common.utility_models.multi_cast_command.MultiCastCommand`)) – The commands to send at specific times
- **vertex_to_send_to** – The vertex these commands are to be sent to

`edges_and_partitions ()`

`generate_data_specification (*args, **kwargs)`

Generate a data specification.

Parameters

- **spec** (`Placement`) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type `None`

`get_binary_file_name ()`

Get the binary name to be run for this vertex.

Return type `str`

Return a string representation of the models binary

`get_binary_start_type ()`

Get the start type of the binary to be run.

Return type `ExecutableType`

`get_edges_and_partitions (pre_vertex, edge_type)`

`static get_n_command_bytes (commands)`

`get_outgoing_partition_constraints (partition)`

Get constraints to be added to the given edge that comes out of this vertex.

Parameters **partition** – An edge that comes out of this vertex

Returns A list of constraints

Return type `list(AbstractConstraint)`

`get_timed_commands_bytes ()`

resources_required

The resources required by the vertex

Return type `ResourceContainer`

class `spinn_front_end_common.utility_models.ChipPowerMonitor` (*label*, *constraints*, *n_samples_per_recording*, *sampling_frequency*)

Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`, `spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`, `spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`

Represents idle time recording code in a application graph.

Parameters

- **label** (*str*) – vertex label
- **constraints** – constraints for the vertex
- **n_samples_per_recording** (*int*) – how many samples to take before recording to SDRAM the total
- **sampling_frequency** – how many microseconds between sampling

create_machine_vertex (*vertex_slice*, *resources_required*, *label=None*, *constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (**args*, ***kwargs*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type `None`

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type `str`

get_binary_start_type ()

Get the start type of the binary to be run.

Return type `ExecutableType`

get_resources_used_by_atoms (**args*, ***kwargs*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type ResourceContainer

Raises None – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type int

class spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex(*args, **kwargs)

Bases: pacman.model.graphs.machine.machine_vertex.MachineVertex, spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary, spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification, spinn_front_end_common.interface.buffer_management.buffer_models.abstract_receive_buffers_to_host.AbstractReceiveBuffersToHost

Machine vertex for C code representing functionality to record idle times in a machine graph.

class CHIP_POWER_MONITOR_REGIONS

Bases: enum.Enum

CONFIG = 1

RECORDING = 2

SYSTEM = 0

SAMPLE_RECORDING_REGION = 0

static binary_file_name()

Return the string binary file name

Returns basestring

static binary_start_type()

The type of binary that implements this vertex

Returns starttype enum

generate_data_specification(*args, **kwargs)

Generate a data specification.

Parameters

- **spec** (Placement) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()

Get the start type of the binary to be run.

Return type ExecutableType

get_recorded_data(placement, buffer_manager)

Get data from SDRAM given placement and buffer manager

Parameters

- **placement** – the location on machine to get data from
- **buffer_manager** – the buffer manager that might have data

Returns results**Return type** numpy array with 1 dimension**get_recorded_region_ids** ()

Get the recording region IDs that have been recorded using buffering

Returns The region numbers that have active recording**Return type** iterable(int)**get_recording_region_base_address** (*txrx, placement*)

Get the recording region base address

Parameters

- **txrx** – the SpiNNMan instance
- **placement** – the placement object of the core to find the address of

Returns the base address of the recording region**static get_resources** (*time_step, time_scale_factor, n_samples_per_recording, sampling_frequency*)

Get the resources used by this vertex

Returns Resource container**n_samples_per_recording****resources_required**

The resources required by the vertex

Return type ResourceContainer**sampling_frequency**

```
class spinn_front_end_common.utility_models.DataSpeedUpPacketGather(x, y,
                                                                    ip_address,
                                                                    ex-
                                                                    tra_monitors_by_chip,
                                                                    re-
                                                                    port_default_directory,
                                                                    write_data_speed_up_reports,
                                                                    con-
                                                                    straints=None)
```

Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`, `spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`, `spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`

create_machine_vertex (*vertex_slice, resources_required, label=None, constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover

- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type *None*

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type *str*

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

machine_vertex

n_atoms

The number of atoms in the vertex

Return type *int*

class `spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex` (*x,*

y,
ex-
tra_monitors,
ip_address,
re-
port_default,
write_data_sp
con-
straints=None

Bases: `pacman.model.graphs.machine.machine_vertex.MachineVertex,`
`spinn_front_end_common.abstract_models.abstract_generates_data_specification.`
`AbstractGeneratesDataSpecification,` `spinn_front_end_common.abstract_models.`
`abstract_has_associated_binary.AbstractHasAssociatedBinary,`
`spinn_front_end_common.interface.provenance.abstract_provides_local_provenance_data.`
`AbstractProvidesLocalProvenanceData`

```

BASE_KEY = 4294967289
BASE_MASK = 4294967291
END_FLAG_KEY = 4294967286
END_FLAG_KEY_OFFSET = 3
FIRST_DATA_KEY = 4294967287
FIRST_DATA_KEY_OFFSET = 2
IN_REPORT_NAME = 'speeds_gained_in_speed_up_process.rpt'
LAST_MESSAGE_FLAG_BIT_MASK = 2147483648
LONG_TIMEOUT = (14, 14)
MISSING_SEQ_NUMS_END_FLAG = 4294967295
NEW_SEQ_KEY = 4294967288
NEW_SEQ_KEY_OFFSET = 1
OUT_REPORT_NAME = 'routers_used_in_speed_up_process.rpt'
SEQUENCE_NUMBER_MASK = 2147483647
SHORT_TIMEOUT = (1, 1)
TEMP_TIMEOUT = (15, 4)
TIMEOUT_PER_RECEIVE_IN_SECONDS = 1
TIME_OUT_FOR_SENDING_IN_SECONDS = 0.01
TRAFFIC_TYPE = 2
TRANSMISSION_THROTTLE_TIME = 1e-06
ZERO_TIMEOUT = (0, 0)

calculate_max_seq_num()
    Deduce the max sequence number expected to be received

    Returns int of the biggest sequence num expected

generate_data_specification(*args, **kwargs)
    Generate a data specification.

    Parameters
    • spec (Placement) – The data specification to write to
    • placement – the placement object this spec is associated with

    Return type None

get_binary_file_name()
    Get the binary name to be run for this vertex.

    Return type str

get_binary_start_type()
    Get the start type of the binary to be run.

    Return type ExecutableType

get_data(placement, memory_address, length_in_bytes, fixed_routes)
    Gets data from a given core and memory address.

```

Parameters

- **placement** – placement object for where to get data from
- **memory_address** – the address in SDRAM to start reading from
- **length_in_bytes** – the length of data to read in bytes
- **fixed_routes** – the fixed routes, used in the report of which chips were used by the speed up process

Returns byte array of the data

get_local_provenance_data()

Get an iterable of provenance data items

Returns iterable of `ProvenanceDataItem`

static load_application_routing_tables (*transceiver, extra_monitor_cores, placements*)

Set all chips to have application table loaded in the router

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type None

static load_system_routing_tables (*transceiver, extra_monitor_cores, placements*)

Set all chips to have the system table loaded in the router

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type None

static locate_correct_write_data_function_for_chip_location (*uses_advanced_monitors, machine, x, y, transceiver, extra_monitor_cores_to_ethernet_connection_map*)

supports other components figuring out which gather and function to call for writing data onto spinnaker

Parameters

- **uses_advanced_monitors** (*bool*) – Whether the system is using advanced monitors
- **machine** – the SpiNNMachine instance
- **x** – the chip x coordinate to write data to
- **y** – the chip y coordinate to write data to
- **extra_monitor_cores_to_ethernet_connection_map** – mapping between cores and connections
- **transceiver** – the SpiNNMan instance

Returns a write function of either a LPG or the spinnMan

Return type func

resources_required

The resources required by the vertex

Return type `ResourceContainer`

send_data_into_spinnaker (*x, y, base_address, data, n_bytes=None, offset=0, cpu=0, is_filename=False*)

sends a block of data into SpiNNaker to a given chip

Parameters

- **x** – chip x for data
- **y** – chip y for data
- **base_address** – the address in SDRAM to start writing memory
- **data** – the data to write
- **n_bytes** – how many bytes to read, or None if not set
- **offset** – where in the data to start from
- **is_filename** (*bool*) – whether data is actually a file.

Return type `None`

set_cores_for_data_streaming (*transceiver, extra_monitor_cores, placements*)

Helper method for setting the router timeouts to a state usable for data streaming

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type `None`

static static_resources_required ()

static streaming (*gatherers, transceiver, extra_monitor_cores, placements*)

Helper method for setting the router timeouts to a state usable for data streaming via a Python context manager (i.e., using the ‘with’ statement).

Parameters

- **gatherers** – All the gatherers that are to be set
- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type a context manager

unset_cores_for_data_streaming (*transceiver, extra_monitor_cores, placements*)

Helper method for setting the router timeouts to a state usable for data streaming

Parameters

- **transceiver** – the SpiNNMan instance
- **extra_monitor_cores** – the extra monitor cores to set
- **placements** – placements object

Return type `None`

class `spinn_front_end_common.utility_models.ExtraMonitorSupport` (*constraints*)
Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`,
`spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`,
`spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`

create_machine_vertex (*vertex_slice, resources_required, label=None, constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable (AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type `None`

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type `str`

get_binary_start_type ()

Get the start type of the binary to be run.

Return type `ExecutableType`

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type `ResourceContainer`

Raises `None` – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type `int`

class `spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex` (*constraints*,
rein-
ject_multicast=Non
rein-
ject_point_to_point
rein-
ject_nearest_neighb
rein-
ject_fixed_route=Fa

Bases: `pacman.model.graphs.machine.machine_vertex.MachineVertex`,
`spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`,
`spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`

Parameters

- **constraints** – constraints on this vertex
- **reinject_multicast** – if we reinject multicast packets; defaults to value of *enable_reinjection* setting in configuration file
- **reinject_point_to_point** – if we reinject point-to-point packets
- **reinject_nearest_neighbour** – if we reinject nearest-neighbour packets
- **reinject_fixed_route** – if we reinject fixed route packets

clear_reinjection_queue (*transceiver*, *placements*, *extra_monitor_cores_to_set*)
 Clears the queues for reinjection

generate_data_specification (**args*, ***kwargs*)
 Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type `None`

get_binary_file_name ()
 Get the binary name to be run for this vertex.

Return type `str`

get_binary_start_type ()
 Get the start type of the binary to be run.

Return type `ExecutableType`

get_reinjection_status (*placements*, *transceiver*)
 Get the reinjection status from this extra monitor vertex

Parameters

- **transceiver** – the spinnMan interface
- **placements** – the placements object

Returns the reinjection status for this vertex

get_reinjection_status_for_vertices (*placements, extra_monitor_cores_for_data, transceiver*)

Get the reinjection status from a set of extra monitor cores

Parameters

- **placements** – the placements object
- **extra_monitor_cores_for_data** – the extra monitor cores to get status from
- **transceiver** – the spinnMan interface

Return type None

load_application_mc_routes (*placements, extra_monitor_cores_for_data, transceiver*)

Get the extra monitor cores to load up the application-based multicast routes (used by data in).

Parameters

- **placements** (*Placements*) – the placements object
- **extra_monitor_cores_for_data** (*iterable(ExtraMonitorSupportMachineVertex)*) – the extra monitor cores to get status from
- **transceiver** (*Transceiver*) – the spinnMan interface

Return type None

load_system_mc_routes (*placements, extra_monitor_cores_for_data, transceiver*)

Get the extra monitor cores to load up the system-based multicast routes (used by data in).

Parameters

- **placements** (*Placements*) – the placements object
- **extra_monitor_cores_for_data** (*iterable(ExtraMonitorSupportMachineVertex)*) – the extra monitor cores to get status from
- **transceiver** (*Transceiver*) – the spinnMan interface

Return type None

placement

reinject_fixed_route

reinject_multicast

reinject_nearest_neighbour

reinject_point_to_point

reset_reinjection_counters (*transceiver, placements, extra_monitor_cores_to_set*)

Resets the counters for reinjection

resources_required

The resources required by the vertex

Return type *ResourceContainer*

set_reinjection_packets (*placements, extra_monitor_cores_for_data, transceiver, point_to_point=None, multicast=None, nearest_neighbour=None, fixed_route=None*)

Parameters

- **placements** – placements object

- **extra_monitor_cores_for_data** – the extra monitor cores to set the packets of
- **transceiver** – spinnman instance
- **point_to_point** (*bool or None*) – If point to point should be set, or None if left as before
- **multicast** (*bool or None*) – If multicast should be set, or None if left as before
- **nearest_neighbour** (*bool or None*) – If nearest neighbour should be set, or None if left as before
- **fixed_route** (*bool or None*) – If fixed route should be set, or None if left as before.

Return type None

set_router_emergency_timeout (*timeout, transceiver, placements, extra_monitor_cores_to_set*)

Sets the timeout of the routers

Parameters

- **timeout** (*(int, int)*) – The mantissa and exponent of the timeout value, each between 0 and 15
- **transceiver** (*Transceiver*) – the spinnMan instance
- **placements** (*Placements*) – the placements object
- **extra_monitor_cores_to_set** – the set of vertices to change the local chip for.

set_router_time_outs (*timeout, transceiver, placements, extra_monitor_cores_to_set*)

Supports setting of the router time outs for a set of chips via their extra monitor cores.

Parameters

- **timeout** (*(int, int)*) – The mantissa and exponent of the timeout value, each between 0 and 15
- **transceiver** (*Transceiver*) – the spinnman interface
- **placements** (*Placements*) – placements object
- **extra_monitor_cores_to_set** – which vertices to use

Return type None

static static_get_binary_file_name()

static static_get_binary_start_type()

static static_resources_required()

```
class spinn_front_end_common.utility_models.LivePacketGather(hostname, port,
                                                             board_address=None,
                                                             tag=None,
                                                             strip_sdp=True,
                                                             use_prefix=False,
                                                             key_prefix=None,
                                                             prefix_type=None,
                                                             mes-
                                                             sage_type=<EIEIOType.KEY_32_BIT:
                                                             2>, right_shift=0,
                                                             pay-
                                                             load_as_time_stamps=True,
                                                             use_payload_prefix=True,
                                                             pay-
                                                             load_prefix=None,
                                                             pay-
                                                             load_right_shift=0,
                                                             num-
                                                             ber_of_packets_sent_per_time_step=0,
                                                             constraints=None,
                                                             label=None)
```

Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`, `spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`, `spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`

A model which stores all the events it receives during a timer tick and then compresses them into Ethernet packets and sends them out of a SpiNNaker machine.

create_machine_vertex (*vertex_slice*, *resources_required*, *label=None*, *constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable (AbstractConstraint)*) – Constraints to be passed on to the machine vertex

generate_data_specification (*spec*, *placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type `None`

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type `str`

get_binary_start_type()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type *int*

```
class spinn_front_end_common.utility_models.LivePacketGatherMachineVertex (label,
                                                                    use_prefix=False,
                                                                    key_prefix=None,
                                                                    pre-
                                                                    fix_type=None,
                                                                    mes-
                                                                    sage_type=<EIEIOType
                                                                    2>,
                                                                    right_shift=0,
                                                                    pay-
                                                                    load_as_time_stamps=True,
                                                                    use_payload_prefix=True,
                                                                    pay-
                                                                    load_prefix=None,
                                                                    pay-
                                                                    load_right_shift=0,
                                                                    num-
                                                                    ber_of_packets_sent_per_tick=1,
                                                                    host-
                                                                    name=None,
                                                                    port=None,
                                                                    strip_sdp=None,
                                                                    board_address=None,
                                                                    tag=None,
                                                                    con-
                                                                    straints=None)
```

Bases: *pacman.model.graphs.machine.machine_vertex.MachineVertex, spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.ProvidesProvenanceDataFromMachineImpl, spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification, spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary, spinn_front_end_common.abstract_models.abstract_supports_database_injection.AbstractSupportsDatabaseInjection*

N_ADDITIONAL_PROVENANCE_ITEMS = 2

TRAFFIC_IDENTIFIER = 'LPG_EVENT_STREAM'

generate_data_specification (*args, **kwargs)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType*

static get_cpu_usage ()

Get the CPU used by this vertex

Returns 0

Return type int

static get_dtcn_usage ()

Get the DTCM used by this vertex

get_provenance_data_from_machine (transceiver, placement)

Get provenance from the machine

Parameters

- **transceiver** – spinnman interface to the machine
- **placement** – the location of this vertex on the machine

static get_sdram_usage ()

Get the SDRAM used by this vertex

is_in_injection_mode

Whether this vertex is actually in injection mode.

resources_required

The resources required by the vertex

Return type *ResourceContainer*

```
class spinn_front_end_common.utility_models.MultiCastCommand(key,          pay-
                                                                load=None,
                                                                time=None,
                                                                repeat=0,      de-
                                                                lay_between_repeats=0)
```

Bases: object

A command to be sent to a vertex.

Parameters

- **key** (*int*) – The key of the command
- **payload** (*int*) – The payload of the command

- **time** (*int*) – The time within the simulation at which to send the command, or None if this is not a timed command
- **repeat** (*int*) – The number of times that the command should be repeated after sending it once. This could be used to ensure that the command is sent despite lost packets. Must be between 0 and 65535
- **delay_between_repeats** (*int*) – The amount of time in microseconds to wait between sending repeats of the same command. Must be between 0 and 65535, and must be 0 if repeat is 0

Raises **SpynnakerException** – If the repeat or delay are out of range

delay_between_repeats

is_payload

Determine if this command has a payload. By default, this returns True if the payload passed in to the constructor is not None, but this can be overridden to indicate that a payload will be generated, despite None being passed to the constructor

Returns True if there is a payload, False otherwise

Return type bool

is_timed

key

payload

Get the payload of the command.

Returns The payload of the command, or None if there is no payload

Return type int

repeat

time

```
class spinn_front_end_common.utility_models.ReverseIpTagMultiCastSource(n_keys,  
                                                                    label=None,  
                                                                    constraints=None,  
                                                                    max_atoms_per_core=922,  
                                                                    board_address=None,  
                                                                    receive_port=None,  
                                                                    receive_sdp_port=1,  
                                                                    receive_tag=None,  
                                                                    receive_rate=10,  
                                                                    virtual_key=None,  
                                                                    pre_fix=None,  
                                                                    pre_fix_type=None,  
                                                                    check_keys=False,  
                                                                    send_buffer_times=None,  
                                                                    send_buffer_partition_id=None,  
                                                                    serve_reverse_ip_tag=False)
```

Bases: `pacman.model.graphs.application.application_vertex.ApplicationVertex`,
`spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`,
`spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`,
`spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints.AbstractProvidesOutgoingPartitionConstraints`,
`spinn_front_end_common.abstract_models.impl.provides_key_to_atom_mapping_impl.ProvidesKeyToAtomMappingImpl`

A model which will allow events to be injected into a SpiNNaker machine and converted into multicast packets.

Parameters

- **n_keys** – The number of keys to be sent via this multicast source
- **label** – The label of this vertex
- **constraints** – Any initial constraints to this vertex
- **board_address** – The IP address of the board on which to place this vertex if receiving data, either buffered or live (by default, any board is chosen)
- **receive_port** – The port on the board that will listen for incoming event packets (default is to disable this feature; set a value to enable it)
- **receive_sdp_port** – The SDP port to listen on for incoming event packets (defaults to 1)
- **receive_tag** – The IP tag to use for receiving live events (uses any by default)
- **receive_rate** – The estimated rate of packets that will be sent by this source
- **virtual_key** – The base multicast key to send received events with (assigned automatically by default)

- **prefix** – The prefix to “or” with generated multicast keys (default is no prefix)
- **prefix_type** – Whether the prefix should apply to the upper or lower half of the multicast keys (default is upper half)
- **check_keys** – True if the keys of received events should be verified before sending (default False)
- **send_buffer_times** – An array of arrays of times at which keys should be sent (one array for each key, default disabled)
- **send_buffer_partition_id** – The ID of the partition containing the edges down which the events are to be sent
- **send_buffer_max_space** – The maximum amount of space to use of the SDRAM on the machine (default is 1MB)
- **send_buffer_space_before_notify** – The amount of space free in the sending buffer before the machine will ask the host for more data (default setting is optimised for most cases)
- **buffer_notification_ip_address** – The IP address of the host that will send new buffers (must be specified if a send buffer is specified or if recording will be used)
- **buffer_notification_port** – The port that the host that will send new buffers is listening on (must be specified if a send buffer is specified, or if recording will be used)
- **buffer_notification_tag** – The IP tag to use to notify the host about space in the buffer (default is to use any tag)

create_machine_vertex (*vertex_slice, resources_required, label=None, constraints=None*)

Create a machine vertex from this application vertex

Parameters

- **vertex_slice** (*Slice*) – The slice of atoms that the machine vertex will cover
- **resources_required** (*ResourceContainer*) – the resources used by the machine vertex
- **label** (*str or None*) – human readable label for the machine vertex
- **constraints** (*iterable(AbstractConstraint)*) – Constraints to be passed on to the machine vertex

enable_recording (*new_state=True*)

generate_data_specification (*spec, placement*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type None

get_binary_file_name ()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type ()

Get the start type of the binary to be run.

Return type *ExecutableType*

get_outgoing_partition_constraints (*partition*)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters **partition** – An edge that comes out of this vertex

Returns A list of constraints

Return type *list(AbstractConstraint)*

get_resources_used_by_atoms (*vertex_slice*)

Get the separate resource requirements for a range of atoms

Parameters **vertex_slice** (*Slice*) – the low value of atoms to calculate resources from

Returns a Resource container that contains a CPUCyclesPerTickResource, DTCMResource and SDRAMResource

Return type *ResourceContainer*

Raises **None** – this method does not raise any known exception

n_atoms

The number of atoms in the vertex

Return type *int*

send_buffer_times

class `spinn_front_end_common.utility_models.ReverseIPTagMulticastSourceMachineVertex` (**args, **kwargs*)

Bases: `pacman.model.graphs.machine.machine_vertex.MachineVertex`,
`spinn_front_end_common.abstract_models.abstract_generates_data_specification.AbstractGeneratesDataSpecification`,
`spinn_front_end_common.abstract_models.abstract_has_associated_binary.AbstractHasAssociatedBinary`,
`spinn_front_end_common.abstract_models.abstract_supports_database_injection.AbstractSupportsDatabaseInjection`,
`spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.ProvidesProvenanceDataFromMachineImpl`,
`spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints.AbstractProvidesOutgoingPartitionConstraints`,
`spinn_front_end_common.interface.buffer_management.buffer_models.sends_buffers_from_host_pre_buffered_impl.SendsBuffersFromHostPreBufferedImpl`,
`spinn_front_end_common.interface.buffer_management.buffer_models.abstract_receive_buffers_to_host.AbstractReceiveBuffersToHost`,
`spinn_front_end_common.abstract_models.abstract_recordable.AbstractRecordable`

A model which allows events to be injected into SpiNNaker and converted in to multicast packets

enable_recording (*new_state=True*)

Enable recording of the keys sent

generate_data_specification (**args, **kwargs*)

Generate a data specification.

Parameters

- **spec** (*Placement*) – The data specification to write to
- **placement** – the placement object this spec is associated with

Return type *None*

get_binary_file_name()

Get the binary name to be run for this vertex.

Return type str

get_binary_start_type()

Get the start type of the binary to be run.

Return type ExecutableType

static get_cpu_usage()

static get_dtcn_usage()

get_outgoing_partition_constraints(partition)

Get constraints to be added to the given edge that comes out of this vertex.

Parameters partition – An edge that comes out of this vertex

Returns A list of constraints

Return type list(AbstractConstraint)

get_provenance_data_from_machine(transceiver, placement)

Get an iterable of provenance data items

Parameters

- **transceiver** – the SpinnMan interface object
- **placement** – the placement of the object

Returns iterable of ProvenanceDataItem

get_recorded_region_ids()

Get the recording region IDs that have been recorded using buffering

Returns The region numbers that have active recording

Return type iterable(int)

get_recording_region_base_address(txrx, placement)

Get the recording region base address

Parameters

- **txrx** – the SpiNNMan instance
- **placement** – the placement object of the core to find the address of

Returns the base address of the recording region

get_region_buffer_size(region)

Get the size of the buffer to be used in SDRAM on the machine for the region in bytes

Parameters region (int) – The region to get the buffer size of

Returns The size of the buffer space in bytes

Return type int

static get_sdram_usage(send_buffer_times, recording_enabled, machine_time_step, receive_rate, n_keys)

is_in_injection_mode

Whether this vertex is actually in injection mode.

is_recording()

Deduce if the recorder is actually recording

mask

static max_send_buffer_keys_per_timestep (*send_buffer_times, n_keys*)

static n_regions_to_allocate (*send_buffering, recording*)

Get the number of regions that will be allocated

static recording_sdram_per_timestep (*machine_time_step, is_recording, receive_rate, send_buffer_times, n_keys*)

resources_required

The resources required by the vertex

Return type [ResourceContainer](#)

static send_buffer_sdram_per_timestep (*send_buffer_times, n_keys*)

send_buffer_times

send_buffers

update_buffer (*arg*)

virtual_key

1.2 Module contents

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

Python Module Index

S

	3
spinn_front_end_common, 148	spinn_front_end_common.abstract_models.impl.provide
spinn_front_end_common.abstract_models,	4
10	spinn_front_end_common.common_model_binaries,
spinn_front_end_common.abstract_models.abstract_can_reset,	13
5	spinn_front_end_common.interface, 54
spinn_front_end_common.abstract_models.abstract_changable_after_run,	spinn_front_end_common.interface.buffer_management,
5	41
spinn_front_end_common.abstract_models.abstract_generates_data_specification,	spinn_front_end_common.interface.buffer_management
6	38
spinn_front_end_common.abstract_models.abstract_has_associated_binary,	spinn_front_end_common.interface.buffer_management
6	16
spinn_front_end_common.abstract_models.abstract_machine_allocation_controller,	spinn_front_end_common.interface.buffer_management
6	13
spinn_front_end_common.abstract_models.abstract_provides_incoming_partition_constraints,	spinn_front_end_common.interface.buffer_management
7	14
spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping,	spinn_front_end_common.interface.buffer_management
7	15
spinn_front_end_common.abstract_models.abstract_provides_n_keys_for_partition,	spinn_front_end_common.interface.buffer_management
7	39
spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints,	spinn_front_end_common.interface.buffer_management
8	28
spinn_front_end_common.abstract_models.abstract_recordable,	spinn_front_end_common.interface.buffer_management
8	18
spinn_front_end_common.abstract_models.abstract_rewrites_data_specification,	spinn_front_end_common.interface.buffer_management
8	19
spinn_front_end_common.abstract_models.abstract_send_me_multicast_commands_vertex,	spinn_front_end_common.interface.buffer_management
8	23
spinn_front_end_common.abstract_models.abstract_supports_database_injection,	spinn_front_end_common.interface.buffer_management
9	25
spinn_front_end_common.abstract_models.abstract_uses_memory_16,	spinn_front_end_common.interface.buffer_management
9	26
spinn_front_end_common.abstract_models.abstract_vertex_with_dependent_vertices,	spinn_front_end_common.interface.buffer_management
9	27
spinn_front_end_common.abstract_models.impl,	spinn_front_end_common.interface.buffer_management
4	27
spinn_front_end_common.abstract_models.impl.machine_allocation_controller,	spinn_front_end_common.interface.config_handler,
3	52
spinn_front_end_common.abstract_models.impl.machine_data_specifiable_vertex,	spinn_front_end_common.interface.java_caller,
	52

spinn_front_end_common.interface.profiling, 68
47 spinn_front_end_common.utilities.notification_prot
spinn_front_end_common.interface.profiling.abstract_has_profile_data, 67
45 spinn_front_end_common.utilities.notification_prot
spinn_front_end_common.interface.profiling.profile_data, 68
45 spinn_front_end_common.utilities.report_functions,
spinn_front_end_common.interface.profiling.profile_utils, 70
46 spinn_front_end_common.utilities.report_functions.k
spinn_front_end_common.interface.provenance, 69
50 spinn_front_end_common.utilities.report_functions.e
spinn_front_end_common.interface.provenance.abstract_provides_local_provenance_data, 69
48 spinn_front_end_common.utilities.report_functions.l
spinn_front_end_common.interface.provenance.abstract_provides_provenance_data_from_machine, 70
49 spinn_front_end_common.utilities.report_functions.m
spinn_front_end_common.interface.provenance.pacman_provenance_extractor, 70
49 spinn_front_end_common.utilities.report_functions.m
spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl, 70
49 spinn_front_end_common.utilities.report_functions.m
spinn_front_end_common.interface.simulation, 70
52 spinn_front_end_common.utilities.scp,
spinn_front_end_common.interface.simulation.simulation_utilities, 72
51 spinn_front_end_common.utilities.scp.clear_iobuf_p
spinn_front_end_common.interface.simulator_state, 71
54 spinn_front_end_common.utilities.scp.scp_clear_iobu
spinn_front_end_common.mapping_algorithms, 71
54 spinn_front_end_common.utilities.scp.scp_update_run
spinn_front_end_common.mapping_algorithms.on_chip_router_table_compression, 72
54 spinn_front_end_common.utilities.scp.update_runtime
spinn_front_end_common.utilities, 102 72
spinn_front_end_common.utilities.connect_spinn, 101
57 spinn_front_end_common.utilities.simulator_interfac
spinn_front_end_common.utilities.connect_spinn_frontend_comment_utilities.utility_objs, 91
55 spinn_front_end_common.utilities.utility_objs.data
spinn_front_end_common.utilities.constants, 86
96 spinn_front_end_common.utilities.utility_objs.dpri
spinn_front_end_common.utilities.database, 87
63 spinn_front_end_common.utilities.utility_objs.execu
spinn_front_end_common.utilities.database_data_base_connection, 87
59 spinn_front_end_common.utilities.utility_objs.execu
spinn_front_end_common.utilities.database_data_base_reader, 87
59 spinn_front_end_common.utilities.utility_objs.execu
spinn_front_end_common.utilities.database_data_base_writer, 88
61 spinn_front_end_common.utilities.utility_objs.extra
spinn_front_end_common.utilities.exceptions, 78
96 spinn_front_end_common.utilities.utility_objs.extra
spinn_front_end_common.utilities.failed_state, 73
97 spinn_front_end_common.utilities.utility_objs.extra
spinn_front_end_common.utilities.globalsspi, 74
97 spinn_front_end_common.utilities.utility_objs.extra
spinn_front_end_common.utilities.helpfulspinn, 74
98 spinn_front_end_common.utilities.utility_objs.extra
spinn_front_end_common.utilities.math_constants, 75
101 spinn_front_end_common.utilities.utility_objs.extra
spinn_front_end_common.utilities.notification_spinn, 75
spinn_front_end_common.utilities.utility_objs.extra

75
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_cpu_packet_size,
 76
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_multicast_port,
 76
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_router_semaphore_port,
 77
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_router_semaphore_port,
 78
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.set_router_semaphore_port,
 78
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes,
 85
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.clear_queue_processes,
 82
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_application,
 83
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_system_mc_r,
 83
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.read_status_processes,
 83
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.reset_counters_processes,
 84
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_packet_types,
 84
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_router_emergencies,
 84
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.set_router_timeout,
 84
 spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters,
 89
 spinn_front_end_common.utilities.utility_objs.provenance_data_item,
 90
 spinn_front_end_common.utilities.utility_objs.reinjection_status,
 90
 spinn_front_end_common.utility_models,
 126
 spinn_front_end_common.utility_models.chip_power_monitor,
 103
 spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex,
 104
 spinn_front_end_common.utility_models.command_sender,
 106
 spinn_front_end_common.utility_models.command_sender_machine_vertex,
 107
 spinn_front_end_common.utility_models.data_speed_up_packet_gatherer,
 109
 spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex,
 110
 spinn_front_end_common.utility_models.extra_monitor_support,
 114
 spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex,
 115
 spinn_front_end_common.utility_models.live_packet_gather,

A

		10
AbstractCanReset	(class in AbstractMachineAllocationController spinn_front_end_common.abstract_models),	(class in spinn_front_end_common.abstract_models.abstract_ma 13 6
AbstractCanReset	(class in AbstractProvidesIncomingPartitionConstraints spinn_front_end_common.abstract_models.abstract_can_reset),	(class in spinn_front_end_common.abstract_models), 5 11
AbstractChangableAfterRun	(class in AbstractProvidesIncomingPartitionConstraints spinn_front_end_common.abstract_models),	(class in spinn_front_end_common.abstract_models.abstract_pro 10 7
AbstractChangableAfterRun	(class in AbstractProvidesKeyToAtomMapping (class in spinn_front_end_common.abstract_models.abstract_changable_after_run),	spinn_front_end_common.abstract_models), 5 11
AbstractDatabase	(class in AbstractProvidesKeyToAtomMapping (class in spinn_front_end_common.interface.buffer_management.storage_objects),	spinn_front_end_common.abstract_models.abstract_provides_key 28 7
AbstractDatabase	(class in AbstractProvidesLocalProvenanceData spinn_front_end_common.interface.buffer_management.storage_objects.abstract_database),	(class in spinn_front_end_common.interface.provenance), 18 50
AbstractGeneratesDataSpecification	(class in AbstractProvidesLocalProvenanceData in spinn_front_end_common.abstract_models),	(class in spinn_front_end_common.interface.provenance.abstract 10 48
AbstractGeneratesDataSpecification	(class in AbstractProvidesNKeysForPartition (class in spinn_front_end_common.abstract_models.abstract_generates_data_specification),	in spinn_front_end_common.abstract_models), 6 11
AbstractHasAssociatedBinary	(class in AbstractProvidesNKeysForPartition (class spinn_front_end_common.abstract_models),	in spinn_front_end_common.abstract_models.abstract_provides_ 10 7
AbstractHasAssociatedBinary	(class in AbstractProvidesOutgoingPartitionConstraints spinn_front_end_common.abstract_models.abstract_has_associated_binary),	(class in spinn_front_end_common.abstract_models), 6 11
AbstractHasProfileData	(class in AbstractProvidesOutgoingPartitionConstraints spinn_front_end_common.interface.profiling),	(class in spinn_front_end_common.abstract_models.abstract_pro 47 8
AbstractHasProfileData	(class in AbstractProvidesProvenanceDataFromMachine spinn_front_end_common.interface.profiling.abstract_has_profile_data),	(class in spinn_front_end_common.interface.provenance), 45 50
AbstractMachineAllocationController	AbstractProvidesProvenanceDataFromMachine (class in spinn_front_end_common.abstract_models),	(class in spinn_front_end_common.interface.provenance.abstract 49

AbstractReceiveBuffersToHost (class in add_commands () (spinn_front_end_common.utility_models.command_s
spinn_front_end_common.interface.buffer_management.buffer_management), 16
16 add_commands () (spinn_front_end_common.utility_models.command_s
AbstractReceiveBuffersToHost (class in method), 107
spinn_front_end_common.interface.buffer_management.buffer_management), (spinn_front_end_common.utility_models.CommandS
13 method), 126
AbstractRecordable (class in add_commands () (spinn_front_end_common.utility_models.CommandS
spinn_front_end_common.abstract_models), method), 128
11 add_data () (spinn_front_end_common.interface.profiling.profile_data.P
AbstractRecordable (class in method), 45
spinn_front_end_common.abstract_models.abstract_recordable), (spinn_front_end_common.interface.profiling.ProfileData
8 method), 47
AbstractRewritesDataSpecification (class add_database_callback ()
in spinn_front_end_common.abstract_models), (spinn_front_end_common.utilities.database.database_connection
12 method), 59
AbstractRewritesDataSpecification (class add_database_callback ()
in spinn_front_end_common.abstract_models.abstract_rewrite_data_specification), (spinn_front_end_common.utilities.database.DatabaseConnection
8 method), 63
AbstractSendMeMulticastCommandsVertex add_init_callback ()
(class in spinn_front_end_common.abstract_models), (spinn_front_end_common.utilities.connections.live_event_conne
12 method), 55
AbstractSendMeMulticastCommandsVertex add_init_callback ()
(class in spinn_front_end_common.abstract_models.abstract_send_me_multicast_commands_vertex), (spinn_front_end_common.utilities.connections.LiveEventConne
8 method), 57
AbstractSendsBuffersFromHost (class in add_key () (spinn_front_end_common.interface.buffer_management.stor
spinn_front_end_common.interface.buffer_management.buffer_management), 16
16 add_key () (spinn_front_end_common.interface.buffer_management.stor
AbstractSendsBuffersFromHost (class in method), 33
spinn_front_end_common.interface.buffer_management.buffer_management), (spinn_front_end_common.interface.buffer_management.stor
14 method), 24
AbstractSupportsDatabaseInjection (class add_keys () (spinn_front_end_common.interface.buffer_management.stor
in spinn_front_end_common.abstract_models), method), 33
12 add_machine_objects ()
AbstractSupportsDatabaseInjection (class (spinn_front_end_common.utilities.database.database_writer.Dat
in spinn_front_end_common.abstract_models.abstract_supports_database_injection),
9 method), 33
add_machine_objects ()
AbstractUsesMemoryIO (class in (spinn_front_end_common.utilities.database.DatabaseWriter
spinn_front_end_common.abstract_models), method), 65
12 add_message_to_send ()
AbstractUsesMemoryIO (class in (spinn_front_end_common.interface.buffer_management.storage_
spinn_front_end_common.abstract_models.abstract_uses_memory_io), 5
9 add_message_to_send ()
AbstractVertexWithEdgeToDependentVertices (spinn_front_end_common.interface.buffer_management.storage_
(class in spinn_front_end_common.abstract_models), method), 34
12 add_pause_stop_callback ()
AbstractVertexWithEdgeToDependentVertices (spinn_front_end_common.utilities.connections.live_event_conne
(class in spinn_front_end_common.abstract_models.abstract_vertex_with_dependent_vertices),
9 method), 34
add_pause_stop_callback ()
add_application_vertices () (spinn_front_end_common.utilities.connections.LiveEventConne
(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter
method), 61
add_placements () (spinn_front_end_common.utilities.database.datab
add_application_vertices () method), 61
(spinn_front_end_common.utilities.database.DatabaseWriter
method), 65
add_placements () (spinn_front_end_common.utilities.database.Datab
method), 66

[add_processor\(\) \(spinn_front_end_common.utilities.utility_objs.SpinnFrontEndCommonUtilities.SimulatorInterface method\), 87](#)
[add_processor\(\) \(spinn_front_end_common.utilities.utility_objs.ExecutableTargets\) \(spinn_front_end_common.utilities.connections.live_event_connection.LiveEventConnection method\), 95](#)
[add_receive_callback\(\) \(spinn_front_end_common.utilities.connections.live_event_connection.LiveEventConnection method\), 55](#)
[add_receive_callback\(\) \(spinn_front_end_common.utilities.connections.LiveEventConnection method\), 57](#)
[add_receive_callback\(\) \(spinn_front_end_common.utilities.connections.live_event_connection.LiveEventConnection method\), 58](#)
[add_receive_label\(\) \(spinn_front_end_common.utilities.connections.live_event_connection.LiveEventConnection method\), 56](#)
[add_receive_label\(\) \(spinn_front_end_common.utilities.connections.LiveEventConnection method\), 58](#)
[add_receive_label\(\) \(spinn_front_end_common.utilities.utility_objs.executable_targets.ExecutableTargets method\), 87](#)
[add_receiving_vertex\(\) \(spinn_front_end_common.interface.buffer_management.buffer_manager.BufferManager method\), 38](#)
[add_receiving_vertex\(\) \(spinn_front_end_common.interface.buffer_management.BufferManager method\), 41](#)
[add_routing_infos\(\) \(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter method\), 62](#)
[add_routing_infos\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[add_routing_tables\(\) \(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter method\), 62](#)
[add_routing_tables\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[add_routing_tables\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[add_send_label\(\) \(spinn_front_end_common.utilities.connections.live_event_connection.LiveEventConnection method\), 56](#)
[add_send_label\(\) \(spinn_front_end_common.utilities.connections.LiveEventConnection method\), 58](#)
[add_sender_vertex\(\) \(spinn_front_end_common.interface.buffer_management.buffer_manager.BufferManager method\), 38](#)
[add_sender_vertex\(\) \(spinn_front_end_common.interface.buffer_management.BufferManager method\), 41](#)
[add_socket_address\(\) \(spinn_front_end_common.utilities.failed_state.FailedState method\), 97](#)
[add_socket_address\(\) \(spinn_front_end_common.utilities.FailedState method\), 102](#)
[add_socket_address\(\) \(spinn_front_end_common.utilities.simulator_interface.SimulatorInterface method\), 101](#)
[add_socket_address\(\) \(spinn_front_end_common.utilities.simulator_interface.SimulatorInterface method\), 107](#)
[add_subsets\(\) \(spinn_front_end_common.utilities.utility_objs.ExecutableTargets method\), 99](#)
[add_system_params\(\) \(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter method\), 62](#)
[add_system_params\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[add_system_params\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[add_tags\(\) \(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter method\), 62](#)
[add_tags\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[add_vertices\(\) \(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter method\), 62](#)
[add_vertices\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[add_vertices\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter method\), 66](#)
[BOARD_REPORT_NAME \(spinn_front_end_common.utilities.report_functions.board_chip_attribute\) attribute\), 99](#)
[auto_detect_database\(\) \(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter static method\), 62](#)
[auto_detect_database\(\) \(spinn_front_end_common.utilities.database.DatabaseWriter static method\), 66](#)
[BASE_KEY \(spinn_front_end_common.utility_models.data_speed_up_packet.DataSpeedUpPacket attribute\), 110](#)
[BASE_KEY \(spinn_front_end_common.utility_models.DataSpeedUpPacket attribute\), 132](#)
[BASE_MASK \(spinn_front_end_common.utility_models.data_speed_up_packet.DataSpeedUpPacket attribute\), 110](#)
[BASE_MASK \(spinn_front_end_common.utility_models.DataSpeedUpPacket attribute\), 133](#)
[BINANT_TILE_NAME \(spinn_front_end_common.utility_models.command_line_arguments.CommandLineArguments attribute\), 107](#)

method), 63	attribute), 130
close () (spinn_front_end_common.utilities.database.DatabaseReader (class in	spinn_front_end_common.interface.config_handler),
method), 64	
close () (spinn_front_end_common.utilities.notification_protocol.notification_protocol.NotificationProtocol	
method), 67	ConfigurationException, 96
close () (spinn_front_end_common.utilities.notification_protocol.NotificationProtocol_chip_and_core_subset ()	
method), 68	(in module spinn_front_end_common.utilities.helpful_functions),
COMMANDS_AT_START_RESUME	98
(spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex (DATA_REGIONS	
attribute), 107	(in module spinn_front_end_common.utilities.helpful_functions),
COMMANDS_AT_START_RESUME	98
(spinn_front_end_common.utility_models.CommandSenderMachineVertex.DATA_REGIONS set () (in	
attribute), 127	module spinn_front_end_common.utilities.helpful_functions),
COMMANDS_AT_STOP_PAUSE	98
(spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex.DATA_REGIONS	
attribute), 107	(spinn_front_end_common.utilities.database.database_writer.DatabaseWriter
COMMANDS_AT_STOP_PAUSE	method), 62
(spinn_front_end_common.utility_models.CommandSenderMachineVertex.DATA_REGIONS	
attribute), 127	(spinn_front_end_common.utilities.database.DatabaseWriter
COMMANDS_WITH_ARBITRARY_TIMES	method), 67
(spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex.DATA_REGIONS	
attribute), 107	(spinn_front_end_common.interface.buffer_management.storage_
COMMANDS_WITH_ARBITRARY_TIMES	static method), 26
(spinn_front_end_common.utility_models.CommandSenderMachineVertex.DATA_REGIONS	
attribute), 128	(spinn_front_end_common.interface.buffer_management.storage_
CommandSender (class in	static method), 35
spinn_front_end_common.utility_models),	create_machine_vertex ()
126	(spinn_front_end_common.utility_models.chip_power_monitor.ChipPowerMonitor
CommandSender (class in	method), 103
spinn_front_end_common.utility_models.command_sender_machine_vertex (	
106	(spinn_front_end_common.utility_models.ChipPowerMonitor
CommandSenderMachineVertex (class in	method), 129
spinn_front_end_common.utility_models),	create_machine_vertex ()
127	(spinn_front_end_common.utility_models.command_sender.CommandSenderMachineVertex
CommandSenderMachineVertex (class in	method), 106
spinn_front_end_common.utility_models.command_sender_machine_vertex (	
107	(spinn_front_end_common.utility_models.CommandSenderMachineVertex
CommandSenderMachineVertex.DATA_REGIONS	method), 126
(class in spinn_front_end_common.utility_models),	create_machine_vertex ()
127	(spinn_front_end_common.utility_models.data_speed_up_packet.DataSpeedUpPacketGenerator
CommandSenderMachineVertex.DATA_REGIONS	method), 109
(class in spinn_front_end_common.utility_models.command_sender_machine_vertex (	
107	(spinn_front_end_common.utility_models.DataSpeedUpPacketGenerator
config (spinn_front_end_common.utilities.failed_state.FailedState method), 131	
attribute), 97	create_machine_vertex ()
config (spinn_front_end_common.utilities.FailedState	(spinn_front_end_common.utility_models.extra_monitor_support.ExtraMonitorSupport
attribute), 102	method), 114
config (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface vertex ()	
attribute), 101	(spinn_front_end_common.utility_models.ExtraMonitorSupport
config (spinn_front_end_common.utilities.SimulatorInterface	method), 136
attribute), 102	create_machine_vertex ()
CONFIG (spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex.ChipPowerMonitorMachineVertex.CHIP_POWER_MONITOR_REGIONS	
attribute), 104	method), 118
CONFIG (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex.CHIP_POWER_MONITOR_REGIONS	

(*spinn_front_end_common.utility_models.LivePacketGather*
 method), 140 DatabaseReader (class in
 create_machine_vertex() *spinn_front_end_common.utilities.database.database_reader*),
 (*spinn_front_end_common.utility_models.reverse_ip_tag_multi_cast_source.ReverseIpTagMultiCastSource*
 method), 123 DatabaseWriter (class in
 create_machine_vertex() *spinn_front_end_common.utilities.database*),
 (*spinn_front_end_common.utility_models.ReverseIpTagMultiCastSource*
 method), 145 DatabaseWriter (class in
 create_schema() (*spinn_front_end_common.utilities.database.database_writer*),
 method), 63 61
 create_schema() (*spinn_front_end_common.utilities.database.database_writer*),
 method), 67 DatabaseWriterGather (class in
 current_read(*spinn_front_end_common.interface.buffer_management.storage_objects.channel_buffer_state.ChannelBufferState*
 attribute), 26 DataSpeedUpPacketGather (class in
 current_read(*spinn_front_end_common.interface.buffer_management.storage_objects.channel_buffer_state*),
 attribute), 35 109
 current_timestamp DataSpeedUpPacketGatherMachineVertex
 (*spinn_front_end_common.interface.buffer_management.storage_objects.channel_buffer_state*), 24
 current_timestamp DataSpeedUpPacketGatherMachineVertex
 (*spinn_front_end_common.interface.buffer_management.storage_objects.channel_buffer_state*), 33
 current_write(*spinn_front_end_common.interface.buffer_management.storage_objects(channel_buffer_state.ChannelBufferState*
 attribute), 26 *spinn_front_end_common.utilities.utility_objs*),
 current_write(*spinn_front_end_common.interface.buffer_management.storage_objects.ChannelBufferState*
 attribute), 35 DataWritten (class in
 cursor(*spinn_front_end_common.utilities.database.database_reader*),
 attribute), 60 86
 cursor(*spinn_front_end_common.utilities.database.DatabaseReader*),
 attribute), 64 delay_between_repeats
 (*spinn_front_end_common.utility_models.multi_cast_command*),
 attribute), 121
D delay_between_repeats
 DATA_IN_COMMANDS (class in (*spinn_front_end_common.utility_models.MultiCastCommand*
spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex),
 110 dependent_vertices()
 data_items(*spinn_front_end_common.interface.provenance.pacman_provenance_extractor.PacmanProvenanceExtractor*
 attribute), 49 method), 9
 data_items(*spinn_front_end_common.interface.provenance.pacman_provenance_extractor.PacmanProvenanceExtractor*
 attribute), 50 (*spinn_front_end_common.abstract_models.AbstractVertexWithE*
 method), 12
 DATA_OUT_COMMANDS (class in *spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex*),
 110 (*spinn_front_end_common.utilities.helpful_functions*),
 database_path(*spinn_front_end_common.utilities.database.database_writer.DatabaseWriter*
 attribute), 63 88
 database_path(*spinn_front_end_common.utilities.database.DatabaseWriter*),
 attribute), 67 (*spinn_front_end_common.interface.provenance.provides_provenance*),
 attribute), 49
 DatabaseConnection (class in DMA_QUEUE_OVERLOADED
spinn_front_end_common.utilities.database), (*spinn_front_end_common.interface.provenance.ProvidesProvenance*
 63 attribute), 51
 DatabaseConnection (class in DPRIFlags (class in
spinn_front_end_common.utilities.database.database_connection), *spinn_front_end_common.utilities.utility_objs*),
 59 91
 DatabaseReader (class in DPRIFlags (class in
spinn_front_end_common.utilities.database), *spinn_front_end_common.utilities.utility_objs.dpri_flags*),

87	END_FLAG_KEY_OFFSET
DURATION (spinn_front_end_common.interface.profiling.profile_data_attribute), 45	ProfileData (class in spinn_front_end_common.utility_models.DataSpeedUpPacketGathererMachineVertex attribute), 133
DURATION (spinn_front_end_common.interface.profiling.ProfileData attribute), 47	EndBufferingState (class in spinn_front_end_common.interface.buffer_management.storage_objects.ChannelBufferState attribute), 36
E	EndBufferingState (class in spinn_front_end_common.interface.buffer_management.storage_objects.ChannelBufferState attribute), 36
edge_partition_identifiers_for_dependent_vertices (spinn_front_end_common.abstract_models.abstract_vertex_with_dependent_vertices.AbstractVertexWithEdgeToDependentVertices method), 9	ENERGY_DETAILED_FILENAME (spinn_front_end_common.utilities.report_functions.energy_report_functions.energy_report_with_data_to_dependent_vertices method), 69
edge_partition_identifiers_for_dependent_vertices (spinn_front_end_common.abstract_models.AbstractVertexWithEdgeToDependentVertices method), 12	ENERGY_DETAILED_FILENAME (spinn_front_end_common.utilities.report_functions.EnergyReportFunctions.energy_report_with_data_to_dependent_vertices method), 70
edges_and_partitions () (spinn_front_end_common.utility_models.command_sender.CommandSender method), 106	ENERGY_SUMMARY_FILENAME (spinn_front_end_common.utilities.report_functions.energy_report_functions.energy_report_with_data_to_dependent_vertices method), 69
edges_and_partitions () (spinn_front_end_common.utility_models.command_sender.CommandSenderMachineVertex method), 108	ENERGY_SUMMARY_FILENAME (spinn_front_end_common.utilities.report_functions.EnergyReportFunctions.energy_report_with_data_to_dependent_vertices method), 70
edges_and_partitions () (spinn_front_end_common.utility_models.CommandSender attribute), 70	EnergyReport (class in spinn_front_end_common.utilities.report_functions), 70
edges_and_partitions () (spinn_front_end_common.utility_models.CommandSenderMachineVertex method), 128	EnergyReport (class in spinn_front_end_common.utilities.report_functions), 70
emergency_recover_state_from_failure () (in module spinn_front_end_common.utilities.helpful_functions), 98	spinn_front_end_common.utilities.report_functions.energy_report_functions.energy_report_with_data_to_dependent_vertices method), 69
emergency_recover_states_from_failure () (in module spinn_front_end_common.utilities.helpful_functions), 99	executable_types_in_binary_set () (spinn_front_end_common.utilities.utility_objs.executable_targets.ExecutableTargets method), 88
enable_recording () (spinn_front_end_common.utility_models.reverse_ip_tag_multi_cast_source.ReverseIpTagMultiCastSource method), 123	executable_types_in_binary_set () (spinn_front_end_common.utilities.utility_objs.ExecutableTargets method), 95
enable_recording () (spinn_front_end_common.utility_models.reverse_ip_tag_multi_cast_source.ReverseIpTagMultiCastSource method), 124	ExecutableFailedToStartException, 96
enable_recording () (spinn_front_end_common.utility_models.reverse_ip_tag_multi_cast_source.ReverseIpTagMultiCastSource method), 145	ExecutableFailedToStopException, 96
enable_recording () (spinn_front_end_common.utility_models.ReverseIpTagMultiCastSource method), 146	ExecutableFinder (class in spinn_front_end_common.utilities.utility_objs), 92
end_address (spinn_front_end_common.interface.buffer_management.storage_objects.ChannelBufferState attribute), 26	ReverseIpTagMultiCastSource (class in spinn_front_end_common.utilities.utility_objs.executable_finder), 87
end_address (spinn_front_end_common.interface.buffer_management.storage_objects.ChannelBufferState attribute), 36	ExecutableNotFoundException, 97
END_FLAG_KEY (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DataSpeedUpPacketGathererMachineVertex attribute), 110	ExecutableTargets (class in spinn_front_end_common.utilities.utility_objs.executable_targets), 87
END_FLAG_KEY (spinn_front_end_common.utility_models.DataSpeedUpPacketGathererMachineVertex attribute), 133	ExecutableType (class in spinn_front_end_common.utilities.utility_objs), 92
END_FLAG_KEY_OFFSET (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DataSpeedUpPacketGathererMachineVertex attribute), 110	

execute_app_data_specification() FIRST_DATA_KEY (spinn_front_end_common.utility_models.DataSpeedUpPacketGathererMachineVertex attribute), 133
 (spinn_front_end_common.interface.java_caller.JavaCaller attribute), 133
 method), 53 FIRST_DATA_KEY_OFFSET
 execute_data_specification() (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DataSpeedUpPacketGathererMachineVertex attribute), 111
 (spinn_front_end_common.interface.java_caller.JavaCaller attribute), 111
 method), 53 FIRST_DATA_KEY_OFFSET
 execute_system_data_specification() (spinn_front_end_common.utility_models.DataSpeedUpPacketGathererMachineVertex attribute), 133
 (spinn_front_end_common.interface.java_caller.JavaCaller attribute), 133
 method), 53 FIXED_ROUTE (spinn_front_end_common.utilities.utility_objs.dpri_flags.DPRIFlags attribute), 87
 EXIT (spinn_front_end_common.utilities.utility_objs.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex attribute), 75
 FIXED_ROUTE (spinn_front_end_common.utilities.utility_objs.DPRIFlags attribute), 91
 extend_allocation() (spinn_front_end_common.abstract_models.abstract_machine_allocation_controller.AbstractMachineAllocationController attribute), 6
 (spinn_front_end_common.utilities.report_functions, 70
 method), 6
 extend_allocation() (spinn_front_end_common.abstract_models.AbstractMachineAllocationControllerReport (class in spinn_front_end_common.utilities.report_functions.fixed_route_flags.FixedRouteFlags attribute), 10
 method), 10
 EXTRA_MONITOR_CORE_DATA_IN_SPEED_UP (spinn_front_end_common.utilities.constants.SDP_PORTS attribute), 96
 EXTRA_MONITOR_CORE_DATA_SPEED_UP (spinn_front_end_common.utilities.constants.SDP_PORTS attribute), 96
 EXTRA_MONITOR_CORE_REINJECTION (spinn_front_end_common.utilities.constants.SDP_PORTS attribute), 96
 extract_provenance() (spinn_front_end_common.interface.provenance.pacman_provenance_extractor.PacmanProvenanceExtractor attribute), 49
 method), 49
G
 extract_provenance() generate_data_specification() (spinn_front_end_common.interface.provenance.PacmanProvenanceExtractor attribute), 50
 (spinn_front_end_common.interface.provenance.pacman_provenance_extractor.PacmanProvenanceExtractor attribute), 50
 method), 50
 ExtraMonitorSupport (class in spinn_front_end_common.utility_models), 136
 generate_data_specification() (spinn_front_end_common.abstract_models.AbstractGeneratesDataSpecification attribute), 10
 method), 10
 ExtraMonitorSupport (class in spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex attribute), 114
 generate_data_specification() (spinn_front_end_common.abstract_models.impl.machine_data_specification.MachineDataSpecification attribute), 3
 method), 3
 ExtraMonitorSupportMachineVertex (class in spinn_front_end_common.utility_models), 136
 generate_data_specification() (spinn_front_end_common.abstract_models.impl.MachineDataSpecification attribute), 4
 method), 4
 ExtraMonitorSupportMachineVertex (class in spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex), 115
 generate_data_specification() (spinn_front_end_common.utility_models.chip_power_monitor.ChipPowerMonitor attribute), 104
 method), 104
F
 FailedState (class in spinn_front_end_common.utilities), 102
 generate_data_specification() (spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex.ChipPowerMonitorMachineVertex attribute), 105
 method), 105
 FailedState (class in spinn_front_end_common.utilities.failed_state), 97
 generate_data_specification() (spinn_front_end_common.utility_models.ChipPowerMonitor attribute), 129
 method), 129
 FINISHED (spinn_front_end_common.interface.simulator_state_simulator_state attribute), 54
 generate_data_specification() (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex attribute), 104
 method), 104
 FIRST_DATA_KEY (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DataSpeedUpPacketGathererMachineVertex attribute), 110
 generate_data_specification()

(spinn_front_end_common.utility_models.command_sender.SpinnFrontEndCommonUtilityModelsCommandSender method), 106	(spinn_front_end_common.utility_models.ReverseIpTagMultiCastSource method), 145
generate_data_specification() (spinn_front_end_common.utility_models.command_sender.SpinnFrontEndCommonUtilityModelsCommandSender method), 108	generate_data_specification() (spinn_front_end_common.utility_models.ReverseIpTagMultiCastSource method), 146
generate_data_specification() (spinn_front_end_common.utility_models.CommandSender (spinn_front_end_common.abstract_models.impl.machine_data_specification method), 126	generate_machine_data_specification() (spinn_front_end_common.abstract_models.impl.machine_data_specification method), 4
generate_data_specification() (spinn_front_end_common.utility_models.CommandSender (spinn_front_end_common.abstract_models.impl.MachineDataSpecification method), 128	generate_machine_data_specification() (spinn_front_end_common.abstract_models.impl.MachineDataSpecification method), 5
generate_data_specification() (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DataSpeedUpPacketGatherMachineVertex method), 109	generate_unique_folder_name() (in module spinn_front_end_common.utilities.database.database_reader.DatabaseReader functions), 99
generate_data_specification() (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DataSpeedUpPacketGatherMachineVertex method), 111	get_all_data() (spinn_front_end_common.interface.java_caller.JavaCaller method), 53
generate_data_specification() (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex method), 132	get_atom_id_to_key_mapping() (spinn_front_end_common.utilities.database.database_reader.DatabaseReader method), 60
generate_data_specification() (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex method), 133	get_atom_id_to_key_mapping() (spinn_front_end_common.utilities.database.DatabaseReader method), 61
generate_data_specification() (spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex method), 114	get_binaries_of_executable_type() (spinn_front_end_common.utilities.utility_objs.executable_targets.ExecutableTargets method), 18
generate_data_specification() (spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex method), 115	get_binaries_of_executable_type() (spinn_front_end_common.utilities.utility_objs.ExecutableTargets method), 19
generate_data_specification() (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex method), 136	get_binary_file_name() (spinn_front_end_common.abstract_models.abstract_has_associated_data.AbstractHasAssociatedData method), 6
generate_data_specification() (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex method), 137	get_binary_file_name() (spinn_front_end_common.abstract_models.AbstractHasAssociatedData method), 7
generate_data_specification() (spinn_front_end_common.utility_models.live_packet_gather_machine_vertex.LivePacketGatherMachineVertex method), 118	get_binary_file_name() (spinn_front_end_common.utility_models.chip_power_monitor.ChipPowerMonitor method), 120
generate_data_specification() (spinn_front_end_common.utility_models.live_packet_gather_machine_vertex.LivePacketGatherMachineVertex method), 120	get_binary_file_name() (spinn_front_end_common.utility_models.chip_power_monitor.ChipPowerMonitor method), 129
generate_data_specification() (spinn_front_end_common.utility_models.LivePacketGatherMachineVertex method), 140	get_binary_file_name() (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex method), 141
generate_data_specification() (spinn_front_end_common.utility_models.LivePacketGatherMachineVertex method), 141	get_binary_file_name() (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex method), 142
generate_data_specification() (spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source.ReverseIpTagMultiCastSource method), 123	get_binary_file_name() (spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex method), 106
generate_data_specification() (spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source.ReverseIpTagMultiCastSource method), 124	get_binary_file_name() (spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex method), 107
generate_data_specification() (spinn_front_end_common.utility_models.CommandSender method), 124	get_binary_file_name() (spinn_front_end_common.utility_models.CommandSender method), 108

<i>method</i>), 127	<i>method</i>), 6
get_binary_file_name() (spinn_front_end_common.utility_models.CommandSenderMachin	get_binary_start_type() spinn_front_end_common.abstract_models.AbstractHasAssociat
<i>method</i>), 128	<i>method</i>), 10
get_binary_file_name() (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer	get_binary_start_type() spinn_front_end_common.utility_models.data_speed_up_packet_gatherer
<i>method</i>), 109	<i>method</i>), 104
get_binary_file_name() (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer	get_binary_start_type() spinn_front_end_common.utility_models.data_speed_up_packet_gatherer
<i>method</i>), 111	<i>method</i>), 105
get_binary_file_name() (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherer	get_binary_start_type() spinn_front_end_common.utility_models.ChipPowerMonitor
<i>method</i>), 132	<i>method</i>), 129
get_binary_file_name() (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherer	get_binary_start_type() spinn_front_end_common.utility_models.ChipPowerMonitorMac
<i>method</i>), 133	<i>method</i>), 130
get_binary_file_name() (spinn_front_end_common.utility_models.extra_monitor_support	get_binary_start_type() spinn_front_end_common.utility_models.command_sender.Com
<i>method</i>), 114	<i>method</i>), 106
get_binary_file_name() (spinn_front_end_common.utility_models.extra_monitor_support	get_binary_start_type() spinn_front_end_common.utility_models.ExtraMonitorSupportMac
<i>method</i>), 115	<i>method</i>), 108
get_binary_file_name() (spinn_front_end_common.utility_models.ExtraMonitorSupport	get_binary_start_type() spinn_front_end_common.utility_models.CommandSender
<i>method</i>), 136	<i>method</i>), 127
get_binary_file_name() (spinn_front_end_common.utility_models.ExtraMonitorSupport	get_binary_start_type() spinn_front_end_common.utility_models.CommandSenderMach
<i>method</i>), 137	<i>method</i>), 128
get_binary_file_name() (spinn_front_end_common.utility_models.live_packet_gatherer	get_binary_start_type() spinn_front_end_common.utility_models.data_speed_up_packet
<i>method</i>), 119	<i>method</i>), 109
get_binary_file_name() (spinn_front_end_common.utility_models.live_packet_gatherer	get_binary_start_type() spinn_front_end_common.utility_models.data_speed_up_packet
<i>method</i>), 120	<i>method</i>), 111
get_binary_file_name() (spinn_front_end_common.utility_models.LivePacketGather	get_binary_start_type() spinn_front_end_common.utility_models.DataSpeedUpPacketGa
<i>method</i>), 140	<i>method</i>), 132
get_binary_file_name() (spinn_front_end_common.utility_models.LivePacketGather	get_binary_start_type() spinn_front_end_common.utility_models.DataSpeedUpPacketGa
<i>method</i>), 142	<i>method</i>), 133
get_binary_file_name() (spinn_front_end_common.utility_models.reverse_ip_tag_multicast	get_binary_start_type() spinn_front_end_common.utility_models.ReverseIPTagMulticastSource
<i>method</i>), 123	<i>method</i>), 114
get_binary_file_name() (spinn_front_end_common.utility_models.reverse_ip_tag_multicast	get_binary_start_type() spinn_front_end_common.utility_models.ReverseIPTagMulticastSource
<i>method</i>), 125	<i>method</i>), 115
get_binary_file_name() (spinn_front_end_common.utility_models.ReverseIPTagMulticast	get_binary_start_type() spinn_front_end_common.utility_models.ExtraMonitorSupport
<i>method</i>), 145	<i>method</i>), 136
get_binary_file_name() (spinn_front_end_common.utility_models.ReverseIPTagMulticast	get_binary_start_type() spinn_front_end_common.utility_models.ExtraMonitorSupportM
<i>method</i>), 146	<i>method</i>), 137
get_binary_start_type() (spinn_front_end_common.abstract_models.abstract_has_associated	get_binary_start_type() spinn_front_end_common.abstract_models.abstract_has_associated

`method`), 60
`get_live_input_details()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 64
`get_live_output_details()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 60
`get_live_output_details()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 64
`get_local_provenance_data()` (`spinn_front_end_common.interface.provenance.abstract_provides_local_provenance` `method`), 48
`get_local_provenance_data()` (`spinn_front_end_common.interface.provenance.AbstractProvidesLocalProvenanceData` `method`), 50
`get_local_provenance_data()` (`spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_interface` `method`), 112
`get_local_provenance_data()` (`spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineInterface` `method`), 134
`get_machine_live_input_details()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 60
`get_machine_live_input_details()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 65
`get_machine_live_input_key()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 61
`get_machine_live_input_key()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 65
`get_machine_live_output_details()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 61
`get_machine_live_output_details()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 65
`get_machine_live_output_key()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 61
`get_machine_live_output_key()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 65
`get_mean_ms()` (`spinn_front_end_common.interface.provenance.abstract_provides_n_keys_for_partition` `method`), 45
`get_mean_ms()` (`spinn_front_end_common.interface.provenance.ProfileData` `method`), 47
`get_mean_ms_per_ts()` (`spinn_front_end_common.interface.provenance.ProfileData` `method`), 45
`get_mean_ms_per_ts()` (`spinn_front_end_common.interface.provenance.ProfileData` `method`), 45
`get_n_atoms()` (`spinn_front_end_common.utilities.database.DatabaseReader` `method`), 48
`get_n_bytes()` (`spinn_front_end_common.interface.buffer_management.storage` `method`), 27
`get_n_calls()` (`spinn_front_end_common.interface.provenance.ProfileData` `method`), 46
`get_n_command_bytes()` (`spinn_front_end_common.utility_models.command_sender_machine_interface` `static method`), 108
`get_n_command_bytes()` (`spinn_front_end_common.utility_models.CommandSenderMachineInterface` `static method`), 128
`get_n_cores_for_executable_type()` (`spinn_front_end_common.utilities.utility_objs.executable_target` `method`), 88
`get_n_cores_for_executable_type()` (`spinn_front_end_common.utilities.utility_objs.ExecutableTargets` `method`), 96
`get_n_keys()` (`spinn_front_end_common.interface.buffer_management` `method`), 24
`get_n_keys()` (`spinn_front_end_common.interface.buffer_management` `method`), 24
`get_n_keys_for_partition()` (`spinn_front_end_common.abstract_models.abstract_provides_n_keys_for_partition` `method`), 7
`get_n_keys_for_partition()` (`spinn_front_end_common.abstract_models.AbstractProvidesNKeysForPartition` `method`), 11
`get_next_key()` (`spinn_front_end_common.interface.buffer_management` `method`), 24

```

        method), 14
get_next_key() (spinn_front_end_common.interface.buffer_management, 16
        method), 16
get_next_key() (spinn_front_end_common.interface.buffer_management, 15
        method), 15
get_next_key() (spinn_front_end_common.interface.buffer_management, 17
        method), 17
get_next_timestamp()
    (spinn_front_end_common.interface.buffer_management.buffer_management, 14
        method), 14
get_next_timestamp()
    (spinn_front_end_common.interface.buffer_management.buffer_management, 16
        method), 16
get_next_timestamp()
    (spinn_front_end_common.interface.buffer_management.buffer_management, 15
        method), 15
get_next_timestamp()
    (spinn_front_end_common.interface.buffer_management.buffer_management, 17
        method), 17
get_not_running_simulator() (in module
    spinn_front_end_common.utilities.globals_variables), 97
get_outgoing_partition_constraints()
    (spinn_front_end_common.abstract_models.abstract_provides_outgoing_partition_constraints, 8
        method), 8
get_outgoing_partition_constraints()
    (spinn_front_end_common.abstract_models.AbstractProvidesOutgoingPartitionConstraints, 11
        method), 11
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.command_sender.CommandSender, 107
        method), 107
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex, 108
        method), 108
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.CommandSender method), 127
        method), 127
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.CommandSenderMachineVertex method), 128
        method), 128
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.ReverseIpTagMulticastSource method), 124
        method), 124
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source.ReverseIpTagMultiCastSource, 125
        method), 125
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.ReverseIpTagMulticastSourceMachineVertex method), 146
        method), 146
get_outgoing_partition_constraints()
    (spinn_front_end_common.utility_models.ReverseIpTagMulticastSourceMachineVertex method), 147
        method), 147
get_placement() (spinn_front_end_common.utilities.database.Database, 61
        method), 61
get_placement() (spinn_front_end_common.utilities.database.Database, 61
        method), 61
get_placements() (spinn_front_end_common.utilities.database.Database, 61
        method), 61
get_placements() (spinn_front_end_common.utilities.database.Database, 61
        method), 61
get_profile_data()
    (spinn_front_end_common.interface.profiling.abstract_has_profile_data, 45
        method), 45
get_profile_data()
    (spinn_front_end_common.interface.profiling.AbstractHasProfileData, 47
        method), 47
get_profile_region_size() (in module
    spinn_front_end_common.interface.profiling.profile_utils), 47
get_profiling_data() (in module
    spinn_front_end_common.interface.profiling.profile_utils), 47
get_provenance_data_from_machine()
    (spinn_front_end_common.interface.provenance.abstract_provides_provenance_data_from_machine, 49
        method), 49
get_provenance_data_from_machine()
    (spinn_front_end_common.interface.provenance.AbstractProvidesProvenanceDataFromMachine, 51
        method), 51
get_provenance_data_from_machine()
    (spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine, 51
        method), 51
get_provenance_data_from_machine()
    (spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachine, 51
        method), 51
get_provenance_data_from_machine()
    (spinn_front_end_common.utility_models.live_packet_gather_machine_vertex.LivePacketGatherMachineVertex, 120
        method), 120
get_provenance_data_from_machine()
    (spinn_front_end_common.utility_models.LivePacketGatherMachineVertex method), 142
        method), 142
get_provenance_data_from_machine()
    (spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source.ReverseIpTagMultiCastSource, 147
        method), 147
get_provenance_data_from_machine()
    (spinn_front_end_common.utility_models.ReverseIpTagMultiCastSourceMachineVertex method), 151
        method), 151
get_provenance_data_size()
    (spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine, 51
        method), 51
get_provenance_data_size()
    (spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachine, 51
        method), 51
get_provenance_data_size()
    (spinn_front_end_common.utility_models.ReverseIpTagMultiCastSourceMachineVertex method), 151
        method), 151
get_recorded_data()
    (spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex.ChipPowerMonitorMachineVertex, 130
        method), 130
get_recorded_data()
    (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex method), 130
        method), 130

```

get_recorded_region_ids() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost method), 13	get_region_buffer_size() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost method), 147
get_recorded_region_ids() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost method), 16	get_region_data() (spinn_front_end_common.interface.buffer_management.storage method), 19
get_recorded_region_ids() (spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex method), 105	get_region_data() (spinn_front_end_common.interface.buffer_management.storage method), 29
get_recorded_region_ids() (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex method), 131	get_region_data() (spinn_front_end_common.interface.buffer_management.storage method), 20
get_recorded_region_ids() (spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex method), 125	get_region_data() (spinn_front_end_common.interface.buffer_management.storage method), 30
get_recorded_region_ids() (spinn_front_end_common.utility_models.ReverseIPTagMulticastSourceMachineVertex method), 147	get_region_data() (spinn_front_end_common.interface.buffer_management.storage method), 27
get_recording_data_constant_size() (in module spinn_front_end_common.interface.buffer_management.utilities), 39	get_region_data() (spinn_front_end_common.interface.buffer_management.storage method), 36
get_recording_header_array() (in module spinn_front_end_common.interface.buffer_management.recording_utilities), 39	get_region_data_pointer() (spinn_front_end_common.interface.buffer_management.storage method), 21
get_recording_header_size() (in module spinn_front_end_common.interface.buffer_management.recording_utilities), 40	get_region_data_pointer() (spinn_front_end_common.interface.buffer_management.storage method), 30
get_recording_region_base_address() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost method), 13	get_region_pointer() (in module spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost), 40
get_recording_region_base_address() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost method), 16	get_regions() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost), 14
get_recording_region_base_address() (spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex method), 105	get_regions() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost), 17
get_recording_region_base_address() (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex method), 131	get_regions() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost), 15
get_recording_region_base_address() (spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex method), 125	get_reinjection_status() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support_utilities), 88
get_recording_region_base_address() (spinn_front_end_common.utility_models.ReverseIPTagMulticastSourceMachineVertex method), 147	get_reinjection_status() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support_utilities), 88
get_region_buffer_size() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost method), 14	get_reinjection_status() (spinn_front_end_common.utility_models.ExtraMonitorSupportUtilities), 88
get_region_buffer_size() (spinn_front_end_common.interface.buffer_management.buffer_models.AbstractReceiveBuffersToHost method), 16	get_reinjection_status_for_core_subsets() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support_utilities), 88
get_region_buffer_size() (spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex method), 125	get_reinjection_status_for_core_subsets() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support_utilities), 88

(spinn_front_end_common.utilities.utility_objs.extra_monitor_responses.ReadStatusProcess
 method), 85
 get_reinjection_status_for_vertices() (spinn_front_end_common.utility_models.extra_monitors.support_machine_vertex.ExtraMonitorSupportMachineVertex
 method), 116
 get_reinjection_status_for_vertices() (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex
 method), 137
 get_resources() (spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex.ChipPowerMonitorMachineVertex
 static method), 105
 get_resources() (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex utilities.utility_objs.extra_monitor_scp
 static method), 131
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.chip_power_monitor_machine_vertex.ChipPowerMonitorMachineVertex
 method), 104
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex utilities.utility_objs.extra_monitor_scp
 method), 129
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.CommandSender (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 method), 107
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.CommandSender (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 method), 127
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.data_speed_up_packet_gather_and_data_speed_up_packet_gather_utilities.utility_objs.extra_monitor_scp
 method), 109
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.DataSpeedUpPacketGather (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 method), 132
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex
 method), 114
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex utilities.utility_objs.extra_monitor_scp
 method), 136
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.live_packet_gather_and_live_packet_gather_utilities.utility_objs.extra_monitor_scp
 method), 119
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.LivePacketGather (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 method), 141
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.reverse_ip_tag_multi_cus_spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 method), 124
 get_resources_used_by_atoms() (spinn_front_end_common.utility_models.ReverseIpTagMultiCusSpinnFrontEndCommonUtilities (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 method), 146
 get_scp_response() (spinn_front_end_common.utilities.scp.scp_clear_iobuf_request_utilities.utility_objs.extra_monitor_scp
 method), 71
 get_scp_response() (spinn_front_end_common.utilities.scp.scp_update_runtime_request_utilities.utility_objs.extra_monitor_scp
 method), 72

get_s dram_usage () (attribute), 102
(spinn_front_end_common.utility_models.live_packet_gather_machine_vertex.LivePacketGatherMachineVertex static method), 120

get_s dram_usage () handle_reinjection_status_response ()
(spinn_front_end_common.utility_models.LivePacketGatherMachineVertex static method), 142 (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.reinjector_scp_commands.ReinjectorSCPCommands static method), 83

get_s dram_usage () handle_reinjection_status_response ()
(spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex.ReverseIPTagMulticastSourceMachineVertex static method), 125 (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.reinjector_scp_commands.ReinjectorSCPCommands static method), 85

get_s dram_usage () has_ran (spinn_front_end_common.utilities.failed_state.FailedState attribute), 142
(spinn_front_end_common.utility_models.ReverseIPTagMulticastSourceMachineVertex static method), 147 (spinn_front_end_common.utilities.failed_state.FailedState attribute), 147

get_simulation_header_array () (in module attribute), 102
spinn_front_end_common.interface.simulation), 52 (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface attribute), 101

get_simulation_header_array () (in module has_ran (spinn_front_end_common.utilities.SimulatorInterface attribute), 102
spinn_front_end_common.interface.simulation.simulation_utilities), 51 (spinn_front_end_common.utilities.SimulatorInterface attribute), 102

get_simulator () (in module has_simulator () (in module
spinn_front_end_common.utilities.globals_variables), 97 spinn_front_end_common.utilities.globals_variables), 97

get_state_for_region () hostname (spinn_front_end_common.utilities.utility_objs.live_packet_gather_machine_vertex.LivePacketGatherMachineVertex attribute), 89
(spinn_front_end_common.interface.buffer_management.storage_objects.EndBufferingState static method), 27 (spinn_front_end_common.utilities.utility_objs.live_packet_gather_machine_vertex.LivePacketGatherMachineVertex attribute), 93

get_state_for_region () increment_none_labelled_vertex_count
(spinn_front_end_common.interface.buffer_management.storage_objects.EndBufferingState static method), 36 (spinn_front_end_common.utilities.failed_state.FailedState attribute), 97

GET_STATUS (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.reinjector_scp_commands.ReinjectorSCPCommands attribute), 75 IN_REPORT_NAME (spinn_front_end_common.utility_models.data_speed.DataSpeed attribute), 11

get_timed_commands_bytes () IN_REPORT_NAME (spinn_front_end_common.utility_models.DataSpeed attribute), 133
(spinn_front_end_common.utility_models.command_sender_machine_vertex.CommandSenderMachineVertex static method), 108 IN_RUN (spinn_front_end_common.interface.simulator_state.SimulatorState attribute), 54

get_timed_commands_bytes () increment_none_labelled_vertex_count
(spinn_front_end_common.utility_models.CommandSenderMachineVertex static method), 128 (spinn_front_end_common.utilities.failed_state.FailedState attribute), 97

GetReinjectionStatusMessage (class in increment_none_labelled_vertex_count
spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages), 78 (spinn_front_end_common.utilities.failed_state.FailedState attribute), 102

GetReinjectionStatusMessage (class in increment_none_labelled_vertex_count
spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.get_reinjection_status_message), 74 (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface attribute), 101

GetReinjectionStatusMessageResponse increment_none_labelled_vertex_count
(class in spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages), 79 (spinn_front_end_common.utilities.SimulatorInterface attribute), 103

GetReinjectionStatusMessageResponse INIT (spinn_front_end_common.interface.simulator_state.SimulatorState attribute), 74
(class in spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_messages.get_reinjection_status_message), 74

graph_mapper (spinn_front_end_common.utilities.failed_state.FailedState attribute), 97 INPUT_BUFFERING_SDP_PORT (spinn_front_end_common.utilities.constants.SDP_PORTS attribute), 96

graph_mapper (spinn_front_end_common.utilities.FailedState attribute), 102 is_data_from_region_flushed ()
(spinn_front_end_common.interface.buffer_management.storage_objects.EndBufferingState static method), 21

graph_mapper (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface attribute), 101 is_data_from_region_flushed ()
(spinn_front_end_common.interface.buffer_management.storage_objects.EndBufferingState static method), 21

graph_mapper (spinn_front_end_common.utilities.SimulatorInterface attribute), 101 is_data_from_region_flushed ()
(spinn_front_end_common.interface.buffer_management.storage_objects.EndBufferingState static method), 21

SpiNNFrontEndCommonEnd Documentation

load_application_mc_routes()	LoadSystemMCRoutesMessage (class in (spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex method), 116	75
load_application_mc_routes()	LoadSystemMCRoutesProcess (class in (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex.ExtraMonitorSupportMachineVertex method), 138	86
load_application_routing_tables()	LoadSystemMCRoutesProcess (class in (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex static method), 112	83
load_application_routing_tables()	locate_correct_write_data_function_for_chip_location() (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex static method), 134	static method), 112
load_initial_buffers()	locate_correct_write_data_function_for_chip_location() (spinn_front_end_common.interface.buffer_management.buffer_manager.BufferManager.utility_models.DataSpeedUpPacketGatherMachineVertex method), 39	static method), 134
load_initial_buffers()	locate_extra_monitor_mc_receiver() (in (spinn_front_end_common.interface.buffer_management.BufferManager.spinn_front_end_common.utilities.helpful_functions), method), 42	100
LOAD_SYSTEM_MC_ROUTES	locate_memory_region_for_placement() (in (spinn_front_end_common.utilities.utility_objs.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex attribute), 78	100
load_system_mc_routes()	LONG_TIMEOUT (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex (spinn_front_end_common.utilities.utility_objs.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex method), 83	LONG_TIMEOUT (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex attribute), 133
load_system_mc_routes()	(spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.LoadSystemMCRoutesProcess method), 86	
load_system_mc_routes()	machine (spinn_front_end_common.utilities.failed_state.FailedState (spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertex method), 116	machine (spinn_front_end_common.utilities.FailedState attribute), 102
load_system_mc_routes()	machine (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex method), 138	machine (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface attribute), 101
load_system_routing_tables()	machine (spinn_front_end_common.utilities.SimulatorInterface (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex static method), 112	machine_time_step
load_system_routing_tables()	(spinn_front_end_common.utilities.failed_state.FailedState (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex static method), 134	machine_time_step
LoadApplicationMCRoutesMessage (class in	(spinn_front_end_common.utilities.FailedState spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_application_mc_routes_message), 81	machine_time_step
LoadApplicationMCRoutesMessage (class in	(spinn_front_end_common.utilities.simulator_interface.SimulatorInterface spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_application_mc_routes_message), 74	machine_time_step
LoadApplicationMCRoutesProcess (class in	(spinn_front_end_common.utilities.SimulatorInterface spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes), 85	machine_time_step
LoadApplicationMCRoutesProcess (class in	machine_vertex (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex attribute), 110	machine_time_step
LoadSystemMCRoutesMessage (class in	MachineAllocationController (class in spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.load_application_mc_routes_message), 82	MachineAllocationController (class in spinn_front_end_common.abstract_models.impl, 4

MachineAllocationController	(class in MemoryMapOnHostReport	(class in
spinn_front_end_common.abstract_models.impl.machine_allocation_controller	spinn_front_end_common.utilities.report_functions.memory_map	
3	70	
MachineDataSpecableVertex	(class in message(spinn_front_end_common.utilities.utility_objs.provenance_data	
spinn_front_end_common.abstract_models.impl),	attribute), 90	
4	message(spinn_front_end_common.utilities.utility_objs.ProvenanceData	
MachineDataSpecableVertex	(class in attribute), 94	
spinn_front_end_common.abstract_models.impl.machine_data_specable_vertex	spinn_front_end_common.utilities.utility_objs.live_pack	
3	attribute), 89	
mark_no_changes()	message_type(spinn_front_end_common.utilities.utility_objs.LivePack	
(spinn_front_end_common.abstract_models.abstract_changable_after_run	AbstractChangableAfterRun	
method), 5	messages(spinn_front_end_common.interface.buffer_management.storage	
mark_no_changes()	attribute), 25	
(spinn_front_end_common.abstract_models.AbstractChangableAfterRun	spinn_front_end_common.interface.buffer_management.storage	
method), 10	attribute), 35	
mark_regions_reloaded()	MILLIWATTS_FOR_BOXED_48_CHIP_FRAME_IDLE_COST	
(spinn_front_end_common.abstract_models.abstract_rewrites_data_specification	AbstractRewritesDataSpecification	
method), 8	attribute), 69	
mark_regions_reloaded()	MILLIWATTS_FOR_BOXED_48_CHIP_FRAME_IDLE_COST	
(spinn_front_end_common.abstract_models.AbstractRewritesDataSpecification	spinn_front_end_common.utilities.report_functions.EnergyReport	
method), 12	attribute), 70	
mask(spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex	ReverseIPTagMulticastSourceMachineVertex	
attribute), 126	(spinn_front_end_common.utilities.report_functions.energy_report	
mask(spinn_front_end_common.utility_models.ReverseIPTagMulticastSourceMachineVertex	attribute), 148	
attribute), 148	MILLIWATTS_FOR_FRAME_IDLE_COST	
MAX_NUMBER_OF_TIMER_TIC_OVERRUN	(spinn_front_end_common.utilities.report_functions.EnergyReport	
(spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl	ProvidesProvenanceDataFromMachineImpl	
attribute), 50	MILLIWATTS_PER_CHIP_ACTIVE_OVERHEAD	
MAX_NUMBER_OF_TIMER_TIC_OVERRUN	(spinn_front_end_common.utilities.report_functions.energy_report	
(spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachineImpl	PROVENANCE_DATA_ENTRIL	
attribute), 51	MILLIWATTS_PER_CHIP_ACTIVE_OVERHEAD	
max_send_buffer_keys_per_timestep()	(spinn_front_end_common.utilities.report_functions.EnergyReport	
(spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex	ReverseIPTagMulticastSourceMachineVertex	
static method), 126	MILLIWATTS_PER_FPGA	
max_send_buffer_keys_per_timestep()	(spinn_front_end_common.utilities.report_functions.energy_report	
(spinn_front_end_common.utility_models.ReverseIPTagMulticastSourceMachineVertex	static method), 148	
static method), 148	MILLIWATTS_PER_FPGA	
memory_used(spinn_front_end_common.utilities.utility_objs.data_written	spinn_front_end_common.utilities.report_functions.EnergyReport	
attribute), 86	attribute), 70	
memory_used(spinn_front_end_common.utilities.utility_objs.DataWritten	PER_FRAME_ACTIVE_COST	
attribute), 91	(spinn_front_end_common.utilities.report_functions.energy_report	
memory_written(spinn_front_end_common.utilities.utility_objs.data_written	DataWritten	
attribute), 86	MILLIWATTS_PER_FRAME_ACTIVE_COST	
memory_written(spinn_front_end_common.utilities.utility_objs.DataWritten	spinn_front_end_common.utilities.report_functions.EnergyReport	
attribute), 91	attribute), 70	
MemoryMapOnHostChipReport	(class in MILLIWATTS_PER_IDLE_CHIP	
spinn_front_end_common.utilities.report_functions),	(spinn_front_end_common.utilities.report_functions.energy_report	
71	attribute), 69	
MemoryMapOnHostChipReport	(class in MILLIWATTS_PER_IDLE_CHIP	
spinn_front_end_common.utilities.report_functions.memory_map_on_host_chip_report	spinn_front_end_common.utilities.report_functions.EnergyReport	
70	attribute), 70	
MemoryMapOnHostReport	(class in MILLIWATTS_PER_UNBOXED_48_CHIP_FRAME_IDLE_COST	
spinn_front_end_common.utilities.report_functions),	(spinn_front_end_common.utilities.report_functions.energy_report	
71	attribute), 69	

(spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex attribute), 131
 n_timestamps (spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region.BufferedSendingRegion attribute), 24
 n_timestamps (spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region.BufferedSendingRegion attribute), 34
 names (spinn_front_end_common.utilities.utility_objs.provenance_data_spinn_front_end_common.utilities.SimulatorInterface attribute), 90
 names (spinn_front_end_common.utilities.utility_objs.ProvenanceDataItemProtocol (class in spinn_front_end_common.utilities.notification_protocol), attribute), 94
 NEAREST_NEIGHBOUR (spinn_front_end_common.utilities.utility_objs.dpriflags.DPRIFlagsProtocol (class in spinn_front_end_common.utilities.notification_protocol.notification attribute), 87
 NEAREST_NEIGHBOUR (spinn_front_end_common.utilities.utility_objs.DPRIFlags_host_name (spinn_front_end_common.utilities.notification_protocol attribute), 91
 NEW_SEQ_KEY (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachine attribute), 111
 NEW_SEQ_KEY (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineMachineVertex attribute), 133
 NEW_SEQ_KEY_OFFSET (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachine attribute), 111
 NEW_SEQ_KEY_OFFSET (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineMachineVertex attribute), 133
 next_key (spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region.BufferedSendingRegion attribute), 24
 next_key (spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region.BufferedSendingRegion attribute), 34
 next_timestamp (spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region.BufferedSendingRegion attribute), 24
 next_timestamp (spinn_front_end_common.interface.buffer_management.storage_objects.buffered_sending_region.BufferedSendingRegion attribute), 34
 NO_APPLICATION (spinn_front_end_common.utilities.utility_objs.executable_type.ExecutableType attribute), 88
 NO_APPLICATION (spinn_front_end_common.utilities.utility_objs.ExecutableType attribute), 92
 no_machine_time_steps (spinn_front_end_common.utilities.failed_state.FailedState attribute), 97
 no_machine_time_steps (spinn_front_end_common.utilities.FailedState attribute), 102
 no_machine_time_steps (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface attribute), 101
 no_machine_time_steps (spinn_front_end_common.utilities.SimulatorInterface attribute), 103
 none_labelled_vertex_count (spinn_front_end_common.utilities.failed_state.FailedState attribute), 97
 none_labelled_vertex_count (spinn_front_end_common.utilities.failed_state.FailedState attribute), 97

partition_id(*spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters*
 attribute), 93 ProfileData (class in
 pause_stop_commands *spinn_front_end_common.interface.profiling.profile_data*),
 (*spinn_front_end_common.abstract_models.abstract_send_multicast_commands_vertex.AbstractSendMeMulticastComm*
 attribute), 8 PROVENANCE_REGION
 pause_stop_commands (*spinn_front_end_common.utility_models.command_sender_machine_commands_vertex*
 (*spinn_front_end_common.abstract_models.AbstractSendMeMulticastCommandsVertex*
 attribute), 12 PROVENANCE_REGION
 payload(*spinn_front_end_common.utility_models.multi_cast_command_sender_machine_commands_vertex.CommandSenderMachine*
 attribute), 121 *attribute*), 128
 payload(*spinn_front_end_common.utility_models.MultiCastCommandDataItem* (class in
 attribute), 143 *spinn_front_end_common.utilities.utility_objs*),
 payload_as_time_stamps 93
 (*spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.LivePacketGatherParameters*
 attribute), 89 *spinn_front_end_common.utilities.utility_objs.provenance_data_i*
 payload_as_time_stamps 90
 (*spinn_front_end_common.utilities.utility_objs.LivePacketGatherParametersMappingImpl* (class in
 attribute), 93 *spinn_front_end_common.abstract_models.impl*),
 payload_prefix(*spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.LivePacketGatherParameters*
 attribute), 89 ProvidesKeyToAtomMappingImpl (class in
 payload_prefix(*spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters*
 attribute), 93 *spinn_front_end_common.abstract_models.impl.provides_key_to*
 4
 payload_right_shift ProvidesProvenanceDataFromMachineImpl
 (*spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.LivePacketGatherParameters*
 attribute), 89 51
 payload_right_shift ProvidesProvenanceDataFromMachineImpl
 (*spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters*
 attribute), 93 49
 placement(*spinn_front_end_common.utility_models.extra_monitor_support_machine_vertex.ExtraMonitorSupportMachineVertexData*
 attribute), 116 (class in *spinn_front_end_common.interface.provenance*),
 placement(*spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertex*
 attribute), 138 ProvidesProvenanceDataFromMachineImpl.PROVENANCE_DA
 placements(*spinn_front_end_common.utilities.failed_state.FailedState* *spinn_front_end_common.interface.provenance.provides*
 attribute), 97 49
 placements(*spinn_front_end_common.utilities.FailedState*
 attribute), 102 R
 placements(*spinn_front_end_common.utilities.simulator_interface.SimulatorInterface*
 attribute), 101 read_config() (in module
 placements(*spinn_front_end_common.utilities.SimulatorInterface**spinn_front_end_common.utilities.helpful_functions*),
 attribute), 103 100
 POINT_TO_POINT(*spinn_front_end_common.utilities.utility_objs.dpriflags.DPRIFlags* (in module
 attribute), 87 *spinn_front_end_common.utilities.helpful_functions*),
 POINT_TO_POINT(*spinn_front_end_common.utilities.utility_objs.DPRIFlags*
 attribute), 91 read_config_int() (in module
 port(*spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.LivePacketGatherParameters*
 attribute), 89 100
 port(*spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters* (in module
 attribute), 93 *spinn_front_end_common.utilities.helpful_functions*),
 prefix_type(*spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.LivePacketGatherParameters*
 attribute), 89 read_data_bytestring()
 prefix_type(*spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters*
 attribute), 93 method), 74
 ProfileData (class in read_data_bytestring()
 spinn_front_end_common.interface.profiling), (*spinn_front_end_common.utilities.utility_objs.extra_monitor_sc*

method), 79

ReadStatusProcess (class in attribute), 116
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes),
 85 (spinn_front_end_common.utility_models.ExtraMonitorSupportM

ReadStatusProcess (class in attribute), 138
 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp_processes.read_status_process),
 83 (spinn_front_end_common.utility_models.extra_monitor_support

RECEIVE_FINISHED (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DATA_IN_COMM
 attribute), 110 reinject_nearest_neighbour

RECEIVE_FIRST_MISSING_SEQ (spinn_front_end_common.utility_models.ExtraMonitorSupportM
 (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DATA_IN_COMMANDS
 attribute), 110 reinject_point_to_point

RECEIVE_MISSING_SEQ_DATA (spinn_front_end_common.utility_models.extra_monitor_support
 (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer_machine_vertex.DATA_IN_COMMANDS
 attribute), 110 reinject_point_to_point

receive_response() (spinn_front_end_common.utility_models.ExtraMonitorSupportM
 (spinn_front_end_common.utilities.scp.clear_iobuf_process_clear_iobuf_process
 method), 71 reinjection_functionality_status

receive_response() (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 (spinn_front_end_common.utilities.scp.ClearIOBUFProcessattribute), 74
 method), 72 reinjection_functionality_status

receive_response() (spinn_front_end_common.utilities.utility_objs.extra_monitor_scp
 (spinn_front_end_common.utilities.scp.update_runtime_process_update_runtime_process
 method), 72 ReInjectionStatus (class in

receive_response() (spinn_front_end_common.utilities.utility_objs),
 (spinn_front_end_common.utilities.scp.UpdateRuntimeProcessattribute), 73
 method), 73 ReInjectionStatus (class in

RECORDING (spinn_front_end_common.utility_models.chip_power_monitoring_machine_vertex.CHIP_POWER_MONITORING
 attribute), 104 90

RECORDING (spinn_front_end_common.utility_models.ChipPowerMonitorMachineVertex.CHIP_POWER_MONITORING
 attribute), 130 spinn_front_end_common.utilities.utility_objs.extra_monitor_scp

recording_sdram_per_timestep() 75
 (spinn_front_end_common.utility_models.reverse_ip_tag_utility.ReverseIPTagUtilityMultiCastCommandM
 static method), 126 attribute), 121

recording_sdram_per_timestep() repeat (spinn_front_end_common.utility_models.MultiCastCommand
 (spinn_front_end_common.utility_models.ReverseIPTagUtilityMachineVertex
 static method), 148 report (spinn_front_end_common.utilities.utility_objs.provenance_data_
 attribute), 90

regenerate_data_specification() (spinn_front_end_common.abstract_models.abstract_rewrite_data_specification.AbstractRewriteDataSpec
 method), 8 attribute), 94

regenerate_data_specification() requires_data_generation
 (spinn_front_end_common.abstract_models.AbstractRewriteDataSpecificationcommon.abstract_models.abstract_changable_
 method), 12 attribute), 6

region_id (spinn_front_end_common.interface.buffer_management_objects.channel_buffer_state.ChannelBufferState
 attribute), 26 (spinn_front_end_common.abstract_models.AbstractChangableA

region_id (spinn_front_end_common.interface.buffer_management_objects.ChannelBufferState
 attribute), 36 requires_mapping (spinn_front_end_common.abstract_models.abstra

reinject_fixed_route (attribute), 6
 (spinn_front_end_common.utility_models.extra_monitor_support_matching_space_file.ExtraMonitorSupportMachineVertexAbstract
 attribute), 116 attribute), 10

reinject_fixed_route requires_memory_regions_to_be_reloaded()
 (spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertexcommon.abstract_models.abstract_rewrites_da
 attribute), 138 method), 8

reinject_multicast requires_memory_regions_to_be_reloaded()

```

(spinn_front_end_common.abstract_models.AbstractRewritesDataSpecificationCommon.utility_models.chip_power_monitor_m
method), 12
attribute), 106
reserve_profile_region() (in module resources_required
spinn_front_end_common.interface.profiling.profile_utils), (spinn_front_end_common.utility_models.ChipPowerMonitorMac
46
attribute), 131
reserve_provenance_data_region() resources_required
(spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_utility_includes_provides_provenance_data_for
method), 50
attribute), 108
reserve_provenance_data_region() resources_required
(spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachineUtilityInIncludesProvidesProvenanceDataFor
method), 51
attribute), 128
reset() (spinn_front_end_common.interface.buffer_management.buffer_manager.BufferManager
method), 39
(spinn_front_end_common.utility_models.data_speed_up_packet
attribute), 142
reset() (spinn_front_end_common.interface.buffer_management.BufferManager) 142
method), 42
resources_required
reset() (spinn_front_end_common.interface.buffer_management.storage_objects.BufferedReceivingData
method), 22
attribute), 135
reset() (spinn_front_end_common.interface.buffer_management.storage_objects.BufferedReceivingData
method), 32
(spinn_front_end_common.utility_models.extra_monitor_support
attribute), 138
RESET_COUNTERS (spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.Reinjector_SCP_commands.Reinjector
attribute), 76
resources_required
reset_counters() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.Reinjector_SCP_commands.Reinjector
method), 82
attribute), 138
reset_counters() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.Reinjector_SCP_commands.Reinjector
method), 85
(spinn_front_end_common.utility_models.live_packet_gather_ma
attribute), 140
reset_counters() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.Reinjector_SCP_commands.Reinjector
method), 84
resources_required
reset_counters() (spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.Reinjector_SCP_commands.Reinjector
method), 85
attribute), 142
reset_reinjection_counters() resources_required
(spinn_front_end_common.utility_models.extra_monitor_support.messages.Reinjector_SCP_commands.Reinjector_SCP_commands.Reinjector
method), 116
attribute), 126
reset_reinjection_counters() resources_required
(spinn_front_end_common.utility_models.ExtraMonitorSupportMachineVertexCommon.utility_models.ReverseIPTagMultiCast
method), 138
attribute), 148
reset_to_first_timestep() resume() (spinn_front_end_common.interface.buffer_management.buffer_manager.BufferManager) 140
(spinn_front_end_common.abstract_models.abstract_can_reset.CanReset
method), 5
resume() (spinn_front_end_common.interface.buffer_management.BufferManager) 142
reset_to_first_timestep() method), 42
(spinn_front_end_common.abstract_models.AbstractCanReset(spinn_front_end_common.interface.buffer_management.storage_objects.BufferedReceivingData
method), 13
method), 22
ResetCountersMessage (class in resume() (spinn_front_end_common.interface.buffer_management.storage_objects.BufferedReceivingData
spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages),
79
ReverseIpTagMultiCastSource (class in
ResetCountersMessage (class in spinn_front_end_common.utility_models), 143
spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.ResetCountersMessage), in
76
spinn_front_end_common.utility_models.reverse_ip_tag_multi_ca
ResetCountersProcess (class in 122
spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.ResetCountersMessage),
85
(class in spinn_front_end_common.utility_models),
ResetCountersProcess (class in 146
spinn_front_end_common.utilities.utility_objs.extra_monitor_support.messages.ResetCountersMessage),
84
(class in spinn_front_end_common.utility_models.reverse_ip_tag_multi_ca
resources_required 124

```

rewind() (spinn_front_end_common.interface.buffer_management.buffer_management_send_buffers_of_function_AbstractSendBuffersOfFunction, 15)
rewind() (spinn_front_end_common.interface.buffer_management.buffer_management_send_buffers_of_function_AbstractSendBuffersOfFunction, 17)
rewind() (spinn_front_end_common.interface.buffer_management.buffer_management_send_buffers_of_function_AbstractSendBuffersOfFunction, 16)
rewind() (spinn_front_end_common.interface.buffer_management.buffer_management_send_buffers_of_function_AbstractSendBuffersOfFunction, 18)
rewind() (spinn_front_end_common.interface.buffer_management.buffer_management_send_buffers_of_function_AbstractSendBuffersOfFunction, 24)
rewind() (spinn_front_end_common.interface.buffer_management.buffer_management_send_buffers_of_function_AbstractSendBuffersOfFunction, 34)
right_shift (spinn_front_end_common.utilities.utility_objs.lvs.pchipgather_and_counters.LivePackedGlobalStateParameters, 89)
right_shift (spinn_front_end_common.utilities.utility_objs.lvs.pchipgather_and_counters.LivePackedGlobalStateParameters, 93)
router_emergency_timeout (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 91)
router_emergency_timeout (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 95)
router_emergency_timeout_parameters (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 96)
router_emergency_timeout_parameters (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 95)
router_timeout (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 91)
router_timeout (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 95)
router_timeout_parameters (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 91)
router_timeout_parameters (spinn_front_end_common.utilities.utility_objs.reinjection_status.ReInjectionStatus, 95)
routing_key_partition_atom_mapping() (spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping.AbstractProvidesKeyToAtomMapping, 7)
routing_key_partition_atom_mapping() (spinn_front_end_common.abstract_models.abstract_provides_key_to_atom_mapping.AbstractProvidesKeyToAtomMapping, 11)
routing_key_partition_atom_mapping() (spinn_front_end_common.abstract_models.impl.provides_key_to_atom_mapping_impl.ProvidesKeyToAtomMappingImpl, 4)
routing_key_partition_atom_mapping() (spinn_front_end_common.abstract_models.impl.provides_key_to_atom_mapping_impl.ProvidesKeyToAtomMappingImpl, 5)
RoutingTableFromMachineReport (class in SDP_PORTS (class in spinn_front_end_common.utilities.report_functions), 71)
RoutingTableFromMachineReport (class in SDP_RUNNING_MESSAGE_CODES (in module spinn_front_end_common.utilities.constants), 96)

182 Index

(module), 49

spinn_front_end_common.interface.provenance (module), 49

spinn_front_end_common.interface.provenance_extractor (module), 70

spinn_front_end_common.interface.provenance_extractor.report_functions (module), 70

spinn_front_end_common.interface.simulator (module), 52

spinn_front_end_common.interface.simulator_utilities (module), 70

spinn_front_end_common.interface.simulator_utilities.scpi (module), 72

spinn_front_end_common.interface.simulator_utilities.scpi.clear_iobuf (module), 71

spinn_front_end_common.mapping_algorithms (module), 54

spinn_front_end_common.mapping_algorithms.spinn_front_end_common.utilities.scpi.scpi_clear_iobuf (module), 71

spinn_front_end_common.mapping_algorithms.spinn_front_end_common.utilities.scpi.scpi_update_run (module), 72

spinn_front_end_common.utilities (module), 102

spinn_front_end_common.utilities.connect (module), 57

spinn_front_end_common.utilities.connect.spinn_front_end_common.utilities.utility_objs (module), 91

spinn_front_end_common.utilities.constants (module), 96

spinn_front_end_common.utilities.database (module), 63

spinn_front_end_common.utilities.database.spinn_front_end_common.utilities.utility_objs.dprint (module), 87

spinn_front_end_common.utilities.database.spinn_front_end_common.utilities.utility_objs.execute (module), 87

spinn_front_end_common.utilities.database.spinn_front_end_common.utilities.utility_objs.execute (module), 87

spinn_front_end_common.utilities.database.spinn_front_end_common.utilities.utility_objs.execute (module), 88

spinn_front_end_common.utilities.exceptions (module), 96

spinn_front_end_common.utilities.failed_state (module), 97

spinn_front_end_common.utilities.global_variables (module), 97

spinn_front_end_common.utilities.helpful_functions (module), 98

spinn_front_end_common.utilities.math_constants (module), 101

spinn_front_end_common.utilities.notifications (module), 68

spinn_front_end_common.utilities.notifications.spinn_front_end_common.utilities.utility_objs.extra (module), 76

spinn_front_end_common.utilities.notifications.spinn_front_end_common.utilities.utility_objs.extra (module), 76

spinn_front_end_common.utilities.notifications.spinn_front_end_common.utilities.utility_objs.extra (module), 76

spinn_front_end_common.utilities.report_functions (module), 70

spinn_front_end_common.utilities.report_functions.spinn_front_end_common.utilities.utility_objs.extra (module), 77

spinn_front_end_common.utilities.report_functions.spinn_front_end_common.utilities.utility_objs.extra (module), 78

spinn_front_end_common.utilities.report_functions.spinn_front_end_common.utilities.utility_objs.extra (module), 78

spinn_front_end_common.utilities.report_functions.spinn_front_end_common.utilities.utility_objs.extra (module), 78

method), 42	static method), 113
stop() (spinn_front_end_common.utilities.failed_state.FailedState method), 97	streaming() (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherParameter, 135
stop() (spinn_front_end_common.utilities.FailedState method), 102	strip_sdp (spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameter.attribute), 89
stop() (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface method), 102	stop() (spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameter.attribute), 93
stop() (spinn_front_end_common.utilities.SimulatorInterface method), 103	stop() (spinn_front_end_common.utilities.utility_objs.executable_type.ExecutableType.attribute), 88
STOP_REQUESTED (spinn_front_end_common.interface.simulator_interface.SimulatorInterface attribute), 54	stop() (spinn_front_end_common.utilities.utility_objs.ExecutableType attribute), 92
store_data_in_region_buffer() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 19	SYSTEM (spinn_front_end_common.utilities.utility_objs.executable_type.ExecutableType.attribute), 88
store_data_in_region_buffer() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 29	SYSTEM (spinn_front_end_common.utilities.utility_objs.ExecutableType attribute), 92
store_data_in_region_buffer() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 22	SYSTEM (spinn_front_end_common.utility_models.ChipPowerMonitorMacros.attribute), 131
store_data_in_region_buffer() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 32	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.command_sequence.attribute), 107
store_data_in_region_buffer() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 28	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_data_in_region_buffer() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 37	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_end_buffering_sequence_number() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 22	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_end_buffering_sequence_number() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 32	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_end_buffering_state() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 22	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_end_buffering_state() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 32	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_last_received_packet_from_core() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 23	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_last_received_packet_from_core() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 32	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_last_sent_packet_to_core() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 23	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
store_last_sent_packet_to_core() (spinn_front_end_common.interface.buffer_management.storage_objects.SQLiteDatabase method), 33	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128
streaming() (spinn_front_end_common.utility_models.data_speed_up_packet_gatherer.unisim_machine_verified_data_speed_up_packet_gatherer.attribute), 111	SYSTEM_BUFFERED_RECEIVING_DATA (spinn_front_end_common.utility_models.CommandSequence.attribute), 128

<i>attribute</i>), 133	TRANSMISSION_EVENT_OVERFLOW
time_scale_factor	(spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachineImpl.PROVENANCE_DATA_ENTRIL
(spinn_front_end_common.utilities.failed_state.FailedState	attribute), 51
<i>attribute</i>), 97	TRANSMISSION_THROTTLE_TIME
time_scale_factor	(spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex
(spinn_front_end_common.utilities.FailedState	attribute), 111
<i>attribute</i>), 102	TRANSMISSION_THROTTLE_TIME
time_scale_factor	(spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex
(spinn_front_end_common.utilities.simulator_interface.SimulatorInterface	<i>attribute</i>), 133
<i>attribute</i>), 102	
time_scale_factor	
(spinn_front_end_common.utilities.SimulatorInterface	
<i>attribute</i>), 103	
timed_commands (spinn_front_end_common.abstract_models.abstract_send_and_receive_multicast_commands_vertex.AbstractSendAndReceiveMulticastCommandsVertex	method), 9
<i>attribute</i>), 9	
timed_commands (spinn_front_end_common.abstract_models.AbstractSendAndReceiveMulticastCommandsVertex	method), 12
<i>attribute</i>), 12	
TIMEOUT_PER_RECEIVE_IN_SECONDS	unset_simulator() (in module
(spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex	<i>attribute</i>), 111
<i>attribute</i>), 111	
TIMEOUT_PER_RECEIVE_IN_SECONDS	update_buffer() (spinn_front_end_common.utility_models.reverse_ip_packet_gather_machine_vertex.ReverseIPPacketGatherMachineVertex
(spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex	<i>attribute</i>), 133
<i>attribute</i>), 133	
TIMER_TIC_HAS_OVERRUN	update_buffer() (spinn_front_end_common.utility_models.ReverseIPPacketGatherMachineVertex
(spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.ProvidesProvenanceDataFromMachineImpl.PROVENANCE_DATA_ENTRIL	<i>attribute</i>), 50
<i>attribute</i>), 50	
TIMER_TIC_HAS_OVERRUN	(spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachineImpl.PROVENANCE_DATA_ENTRIL
(spinn_front_end_common.interface.provenance.ProvidesProvenanceDataFromMachineImpl.PROVENANCE_DATA_ENTRIL	<i>attribute</i>), 51
<i>attribute</i>), 51	
timestamps (spinn_front_end_common.interface.buffer_management.storage.objects.buffered_sending_region.BufferedSendingRegion	<i>attribute</i>), 24
<i>attribute</i>), 24	
timestamps (spinn_front_end_common.interface.buffer_management.storage.objects.BufferedSendingRegion	<i>attribute</i>), 34
<i>attribute</i>), 34	
TRAFFIC_IDENTIFIER	update_last_received_sequence_number() (spinn_front_end_common.interface.buffer_management.storage.objects.buffered_sending_region.BufferedSendingRegion
(spinn_front_end_common.utility_models.live_packet_gather_machine_vertex.LivePacketGatherMachineVertex	<i>attribute</i>), 120
<i>attribute</i>), 120	
TRAFFIC_IDENTIFIER	update_last_received_sequence_number() (spinn_front_end_common.interface.buffer_management.storage.objects.buffered_sending_region.BufferedSendingRegion
(spinn_front_end_common.utility_models.LivePacketGatherMachineVertex	<i>attribute</i>), 141
<i>attribute</i>), 141	
TRAFFIC_TYPE (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex	<i>attribute</i>), 111
<i>attribute</i>), 111	
TRAFFIC_TYPE (spinn_front_end_common.utility_models.DataSpeedUpPacketGatherMachineVertex	<i>attribute</i>), 133
<i>attribute</i>), 133	
transceiver (spinn_front_end_common.utilities.failed_state.FailedState	<i>attribute</i>), 97
<i>attribute</i>), 97	
transceiver (spinn_front_end_common.utilities.FailedState	<i>attribute</i>), 102
<i>attribute</i>), 102	
transceiver (spinn_front_end_common.utilities.simulator_interface.SimulatorInterface	<i>attribute</i>), 102
<i>attribute</i>), 102	
transceiver (spinn_front_end_common.utilities.SimulatorInterface	<i>attribute</i>), 103
<i>attribute</i>), 103	
TRANSMISSION_EVENT_OVERFLOW	(spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.ProvidesProvenanceDataFromMachineImpl.PROVENANCE_DATA_ENTRIL
(spinn_front_end_common.interface.provenance.provides_provenance_data_from_machine_impl.ProvidesProvenanceDataFromMachineImpl.PROVENANCE_DATA_ENTRIL	<i>attribute</i>), 50
<i>attribute</i>), 50	

U

unset_cores_for_data_streaming() (spinn_front_end_common.utility_models.data_speed_up_packet_gather_machine_vertex.DataSpeedUpPacketGatherMachineVertex	method), 135
unset_cores_for_data_streaming() (spinn_front_end_common.utility_models.ReverseIPPacketGatherMachineVertex	method), 148
update_buffer() (spinn_front_end_common.utility_models.reverse_ip_packet_gather_machine_vertex.ReverseIPPacketGatherMachineVertex	<i>attribute</i>), 133
<i>attribute</i>), 133	
update_buffer() (spinn_front_end_common.utility_models.ReverseIPPacketGatherMachineVertex	<i>attribute</i>), 148
<i>attribute</i>), 148	
update_read_pointer() (spinn_front_end_common.interface.buffer_management.storage.objects.buffered_sending_region.BufferedSendingRegion	<i>attribute</i>), 26
<i>attribute</i>), 26	
update_runtime() (spinn_front_end_common.utilities.scp.update_runtime() (in module	method), 72
update_runtime() (spinn_front_end_common.utilities.scp.UpdateRuntime() (in module	method), 73
update_sequence_no_for_core() (spinn_front_end_common.interface.buffer_management.storage.objects.buffered_sending_region.BufferedSendingRegion	<i>attribute</i>), 23
<i>attribute</i>), 23	
update_sequence_no_for_core() (spinn_front_end_common.interface.buffer_management.storage.objects.buffered_sending_region.BufferedSendingRegion	<i>attribute</i>), 33
<i>attribute</i>), 33	
update_runtime() (spinn_front_end_common.utilities.scp.update_runtime() (in module	method), 73
update_runtime() (spinn_front_end_common.utilities.scp.UpdateRuntime() (in module	method), 73

UpdateRuntimeProcess (class in W
spinn_front_end_common.utilities.scp.update_runtime_process)
72 wait_for_confirmation()
use_payload_prefix (spinn_front_end_common.utilities.notification_protocol.notification
method), 67
(spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.LivePacketGatherParameters
attribute), 89 wait_for_confirmation()
use_payload_prefix (spinn_front_end_common.utilities.notification_protocol.Notification
method), 68
(spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters
attribute), 93 write_address_to_user0() (in module
spinn_front_end_common.utilities.helpful_functions),
use_prefix(spinn_front_end_common.utilities.utility_objs.live_packet_gather_parameters.LivePacketGatherParameters
attribute), 89 101
use_prefix(spinn_front_end_common.utilities.utility_objs.LivePacketGatherParameters
attribute), 93 write_data_to_memory_io()
use_virtual_board (spinn_front_end_common.abstract_models.abstract_uses_memory
method), 9
(spinn_front_end_common.utilities.failed_state.FailedState (spinn_front_end_common.abstract_models.AbstractUsesMemory
attribute), 97 method), 13
use_virtual_board write_finished_file()
(spinn_front_end_common.utilities.FailedState (spinn_front_end_common.interface.config_handler.ConfigHandler
attribute), 102 method), 52
use_virtual_board write_profile_region_data() (in module
(spinn_front_end_common.utilities.simulator_interface.SimulatorInterface spinn_front_end_common.interface.profiling.profile_utils),
attribute), 102 47
use_virtual_board (spinn_front_end_common.utilities.SimulatorInterface
attribute), 103 Z
USES_SIMULATION_INTERFACE ZERO_TIMEOUT(spinn_front_end_common.utility_models.data_speed_up
attribute), 111
(spinn_front_end_common.utilities.utility_objs.executable_type.ExecutableType
attribute), 88 ZERO_TIMEOUT(spinn_front_end_common.utility_models.DataSpeedUp
attribute), 133
USES_SIMULATION_INTERFACE
(spinn_front_end_common.utilities.utility_objs.ExecutableType
attribute), 92

V

value(spinn_front_end_common.utilities.utility_objs.provenance_data_item.ProvenanceDataItem
attribute), 90
value(spinn_front_end_common.utilities.utility_objs.ProvenanceDataItem
attribute), 94
verify_not_running()
(spinn_front_end_common.utilities.failed_state.FailedState
method), 97
verify_not_running()
(spinn_front_end_common.utilities.FailedState
method), 102
verify_not_running()
(spinn_front_end_common.utilities.simulator_interface.SimulatorInterface
method), 102
verify_not_running()
(spinn_front_end_common.utilities.SimulatorInterface
method), 103
virtual_key(spinn_front_end_common.utility_models.reverse_ip_tag_multicast_source_machine_vertex.ReverseIPTagMulticastSourceMachineVertex
attribute), 126
virtual_key(spinn_front_end_common.utility_models.ReverseIPTagMulticastSourceMachineVertex
attribute), 148