

# Punnett's Reference Manual

1.0a

Generated by Doxygen 1.2.18

Mon Jun 16 15:42:24 2003



# Contents

<b>1</b>	<b>Punnets Main Page</b>	<b>1</b>
<b>2</b>	<b>Punnets Module Index</b>	<b>3</b>
2.1	Punnets Modules . . . . .	3
<b>3</b>	<b>Punnets Namespace Index</b>	<b>5</b>
3.1	Punnets Namespace List . . . . .	5
<b>4</b>	<b>Punnets Hierarchical Index</b>	<b>7</b>
4.1	Punnets Class Hierarchy . . . . .	7
<b>5</b>	<b>Punnets Compound Index</b>	<b>9</b>
5.1	Punnets Compound List . . . . .	9
<b>6</b>	<b>Punnets File Index</b>	<b>11</b>
6.1	Punnets File List . . . . .	11
<b>7</b>	<b>Punnets Page Index</b>	<b>13</b>
7.1	Punnets Related Pages . . . . .	13
<b>8</b>	<b>Punnets Module Documentation</b>	<b>15</b>
8.1	Logging (drawing a graph of neuron potentials) . . . . .	15
8.2	Variables for Statistics . . . . .	16
8.3	Neurons . . . . .	18
8.4	Synapses . . . . .	19
8.5	Scheduling . . . . .	20
<b>9</b>	<b>Punnets Namespace Documentation</b>	<b>21</b>
9.1	punnets Namespace Reference . . . . .	21
9.2	punnets_common Namespace Reference . . . . .	23
9.3	punnets_nodebug Namespace Reference . . . . .	29

9.4	punnets_private Namespace Reference . . . . .	30
<b>10</b>	<b>Punnets Class Documentation</b>	<b>31</b>
10.1	punnets_common::debugflag< false > Class Template Reference . . . . .	31
10.2	punnets_common::debugflag< true > Class Template Reference . . . . .	32
10.3	punnets_common::func_base Class Reference . . . . .	33
10.4	punnets_common::func_const Class Reference . . . . .	36
10.5	punnets_common::func_const_int Class Reference . . . . .	38
10.6	punnets_common::func_delta_int Class Reference . . . . .	40
10.7	punnets_common::func_delta_int::message_add_pulse Class Reference . . . . .	42
10.8	punnets_common::func_deriveq_base Class Reference . . . . .	43
10.9	punnets_common::func_deriveq_base::message_set_lambda Class Reference . . . . .	44
10.10	punnets_common::func_deriveq_base::message_set_zero_point Class Reference . . . . .	45
10.11	punnets_common::func_exp Class Reference . . . . .	46
10.12	punnets_common::func_exp_diff Class Reference . . . . .	48
10.13	punnets_common::func_exp_diff::message_add_event_time Class Reference . . . . .	50
10.14	punnets_common::func_exp_int Class Reference . . . . .	51
10.15	punnets_common::func_response Class Reference . . . . .	53
10.16	punnets_common::func_sine Class Reference . . . . .	55
10.17	punnets_common::func_sine_int Class Reference . . . . .	57
10.18	punnets_common::func_sineshot Class Reference . . . . .	59
10.19	punnets_common::func_sineshot::message_set_t0 Class Reference . . . . .	62
10.20	punnets_common::func_sineshot_int Class Reference . . . . .	63
10.21	punnets_common::func_step Class Reference . . . . .	65
10.22	punnets_common::greater_tevent Struct Reference . . . . .	67
10.23	punnets_common::message_base Class Reference . . . . .	68
10.24	punnets_common::taction Class Reference . . . . .	69
10.25	punnets_common::tevent Class Reference . . . . .	71
10.26	punnets_common::tlogger Class Reference . . . . .	72
10.27	punnets_private::tneuron< debug > Class Template Reference . . . . .	76
10.28	punnets_common::tneuron_base Class Reference . . . . .	79
10.29	punnets_private::tneuron_ext< debug > Class Template Reference . . . . .	81
10.30	punnets_private::tneuron_ext_const< debug > Class Template Reference . . . . .	84
10.31	tobserver Class Reference . . . . .	87
10.32	punnets_common::tsched_double Class Reference . . . . .	88
10.33	punnets_common::tsentinel Class Reference . . . . .	89
10.34	punnets_private::tsynapse< debug > Class Template Reference . . . . .	90

---

10.35	punnets_private::tsynapse_addfunc< debug > Class Template Reference . . . . .	92
10.36	punnets_common::tsynapse_base Class Reference . . . . .	94
10.37	punnets_private::tsynapse_fatigue< debug > Class Template Reference . . . . .	96
10.38	punnets_private::tsynapse_message< debug > Class Template Reference . . . . .	98
10.39	punnets_private::tsynapse_messfunc< debug > Class Template Reference . . . . .	100
<b>11</b>	<b>Punnets File Documentation</b>	<b>103</b>
11.1	dlanguage.cpp File Reference . . . . .	103
11.2	dlogger.cpp File Reference . . . . .	105
11.3	dlogger.h File Reference . . . . .	106
11.4	dneuron.cpp File Reference . . . . .	107
11.5	dneuron.h File Reference . . . . .	108
11.6	dsched.cpp File Reference . . . . .	109
11.7	dsched.h File Reference . . . . .	110
11.8	dtest.cpp File Reference . . . . .	111
11.9	func.cpp File Reference . . . . .	112
11.10	func.h File Reference . . . . .	113
11.11	punnets.h File Reference . . . . .	114
11.12	punnets_base.h File Reference . . . . .	115
<b>12</b>	<b>Punnets Page Documentation</b>	<b>117</b>
12.1	Todo List . . . . .	117



# Chapter 1

## Punnets Main Page

Punnets is a C++ library for Pulsed Neural Network Simulator. Based on its event-discrete (event-driven) manner of simulation, Punnets provides highly accurate and efficient simulation environment for rather complex neuron models. Please take a look at the sample programs, **dtest.cpp** and **dlanguage.cpp**, to see how to use the library.

Punnets provides two user-accessible namespaces, `punnets` and **punnets\_nodebug** (p. 29). A user program may import one of the two namespaces to use the library. Although the **punnets\_nodebug** (p. 29) version is more efficient in simulation, `punnets` version can provide debugging facility, which can be turned on for neuron-by-neuron basis. Stuffs in **punnets\_common** (p. 23) namespace can be accessed via either namespace, `punnets` or **punnets\_nodebug** (p. 29).

You can obtain the reference manual of this library by using doxygen, a document generation tool. If you have installed doxygen, it will be automatically generated and installed via a standard install process. You can also see the reference manual via the Punnets home page: <http://www.snowelm.com/~t/research/software/punnets/>





# Chapter 2

## Punnets Module Index

### 2.1 Punnets Modules

Here is a list of all modules:

Logging (drawing a graph of neuron potentials) . . . . .	15
Variables for Statistics . . . . .	16
Neurons . . . . .	18
Synapses . . . . .	19
Scheduling . . . . .	20



# Chapter 3

## Punnets Namespace Index

### 3.1 Punnets Namespace List

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

<b>punnets</b> . . . . .	21
<b>punnets_common</b> (This namespace provides commonly available classes. Users should use classes and functions in this namespace via either punnets or <b>punnets_nodebug</b> (p. 29) namespace) . . . . .	23
<b>punnets_nodebug</b> . . . . .	29
<b>punnets_private</b> (This namespace provides some private classes used for punnets library. Users should not use classes and functions in this namespace directly) . . . . .	30



# Chapter 4

## Punnets Hierarchical Index

### 4.1 Punnets Class Hierarchy

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

punnets_common::debugflag< false > . . . . .	31
punnets_common::debugflag< true > . . . . .	32
punnets_common::func_base . . . . .	33
punnets_common::func_const . . . . .	36
punnets_common::func_deriveq_base . . . . .	43
punnets_common::func_const_int . . . . .	38
punnets_common::func_delta_int . . . . .	40
punnets_common::func_exp_int . . . . .	51
punnets_common::func_response . . . . .	53
punnets_common::func_sine_int . . . . .	57
punnets_common::func_sineshot_int . . . . .	63
punnets_common::func_exp . . . . .	46
punnets_common::func_exp_diff . . . . .	48
punnets_common::func_sine . . . . .	55
punnets_common::func_sineshot . . . . .	59
punnets_common::func_step . . . . .	65
punnets_common::greater_tevent . . . . .	67
punnets_common::message_base . . . . .	68
punnets_common::func_delta_int::message_add_pulse . . . . .	42
punnets_common::func_deriveq_base::message_set_lambda . . . . .	44
punnets_common::func_deriveq_base::message_set_zero_point . . . . .	45
punnets_common::func_exp_diff::message_add_event_time . . . . .	50
punnets_common::func_sineshot::message_set_t0 . . . . .	62
neuinfo	
punnets_common::taction . . . . .	69
punnets_common::tlogger . . . . .	72
punnets_common::tmessage	
punnets_common::tpulse	
punnets_common::tsentinel . . . . .	89
punnets_common::tsynapse_base . . . . .	94
punnets_private::tsynapse< debug > . . . . .	90
punnets_private::tsynapse_addfunc< debug > . . . . .	92

---

punnets_private::tsynapse_fatigue< debug > . . . . .	96
punnets_private::tsynapse_message< debug > . . . . .	98
punnets_private::tsynapse_messfunc< debug > . . . . .	100
punnets_private::tneuron< debug > . . . . .	76
punnets_private::tneuron_ext_const< debug > . . . . .	84
punnets_private::tneuron_ext< debug > . . . . .	81
punnets_common::tevent . . . . .	71
punnets_common::tneuron_base . . . . .	79
punnets_private::tneuron< debug > . . . . .	76
punnets_private::tneuron_ext< debug > . . . . .	81
tobserver . . . . .	87
punnets_common::tsched_double . . . . .	88
punnets_private::valdomain	
word_t	

# Chapter 5

## Punnets Compound Index

### 5.1 Punnets Compound List

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

<code>punnets_common::debugflag&lt; false &gt;</code> . . . . .	31
<code>punnets_common::debugflag&lt; true &gt;</code> (This instantiation of the class is used for punnets namespace) . . . . .	32
<code>punnets_common::func_base</code> . . . . .	33
<code>punnets_common::func_const</code> (</classdef>) . . . . .	36
<code>punnets_common::func_const_int</code> . . . . .	38
<code>punnets_common::func_delta_int</code> (</classdef>) . . . . .	40
<code>punnets_common::func_delta_int::message_add_pulse</code> (A message that adds a new pulse) . . . . .	42
<code>punnets_common::func_deriveq_base</code> . . . . .	43
<code>punnets_common::func_deriveq_base::message_set_lambda</code> (A message that changes lambda (leak value)) . . . . .	44
<code>punnets_common::func_deriveq_base::message_set_zero_point</code> (A message that changes zero point) . . . . .	45
<code>punnets_common::func_exp</code> . . . . .	46
<code>punnets_common::func_exp_diff</code> . . . . .	48
<code>punnets_common::func_exp_diff::message_add_event_time</code> . . . . .	50
<code>punnets_common::func_exp_int</code> . . . . .	51
<code>punnets_common::func_response</code> . . . . .	53
<code>punnets_common::func_sine</code> . . . . .	55
<code>punnets_common::func_sine_int</code> . . . . .	57
<code>punnets_common::func_sineshot</code> . . . . .	59
<code>punnets_common::func_sineshot::message_set_t0</code> (A message that changes t0) . . . . .	62
<code>punnets_common::func_sineshot_int</code> . . . . .	63
<code>punnets_common::func_step</code> (</classdef>) . . . . .	65
<code>punnets_common::greater_tevent</code> . . . . .	67
<code>punnets_common::message_base</code> . . . . .	68
<code>punnets_common::taction</code> . . . . .	69
<code>punnets_common::tevent</code> . . . . .	71
<code>punnets_common::tlogger</code> . . . . .	72
<code>punnets_private::tneuron&lt; debug &gt;</code> . . . . .	76
<code>punnets_common::tneuron_base</code> . . . . .	79
<code>punnets_private::tneuron_ext&lt; debug &gt;</code> . . . . .	81

---

<code>punnets_private::tneuron_ext_const</code> < <code>debug</code> > . . . . .	84
<code>tobserver</code> . . . . .	87
<code>punnets_common::tsched_double</code> . . . . .	88
<code>punnets_common::tsentinel</code> . . . . .	89
<code>punnets_private::tsynapse</code> < <code>debug</code> > . . . . .	90
<code>punnets_private::tsynapse_addfunc</code> < <code>debug</code> > (Synapse class that adds a new function to the destination (postsynaptic) <code>tneuron_ext</code> (p. 81)) . . . . .	92
<code>punnets_common::tsynapse_base</code> . . . . .	94
<code>punnets_private::tsynapse_fatigue</code> < <code>debug</code> > . . . . .	96
<code>punnets_private::tsynapse_message</code> < <code>debug</code> > . . . . .	98
<code>punnets_private::tsynapse_messfunc</code> < <code>debug</code> > . . . . .	100



# Chapter 6

## Punnets File Index

### 6.1 Punnets File List

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

<b>config_punnets.h</b>	??
<b>dlanguage.cpp</b> (Language simulation based on the punnets library)	103
<b>dlogger.cpp</b> (Logger)	105
<b>dlogger.h</b> (Activation logging class)	106
<b>dneuron.cpp</b> (Neurons)	107
<b>dneuron.h</b> (Neuron/Synapse class in discrete-event NN simulation)	108
<b>dsched.cpp</b> (Event, Action, Scheduler)	109
<b>dsched.h</b> (Distributed scheduler)	110
<b>dtest.cpp</b> (Test program of the punnets library)	111
<b>func.cpp</b> (Functions)	112
<b>func.h</b> (Function representation)	113
<b>punnets.h</b> (Punnets header file)	114
<b>punnets_base.h</b> (Base class of Punnets)	115



# Chapter 7

## Punnets Page Index

### 7.1 Punnets Related Pages

Here is a list of all related documentation pages:

Todo List . . . . .	117
---------------------	-----



## Chapter 8

# Punnets Module Documentation

### 8.1 Logging (drawing a graph of neuron potentials)

#### Compounds

- class `tlogger`

## 8.2 Variables for Statistics

### Variables

- `u_int64_t totalfire = 0`

*This variable counts the total number of firing of the neurons.*

- `u_int64_t totalpulse = 0`

*This variable counts the total pulse arrivals to any of the neurons.*

- `u_int64_t totalpartition = 0`

*This variable counts the total number of partitions.*

- `u_int64_t totalpartition_nonewton [4] = {0,0,0,0}`

- `u_int64_t totalpartition_newton [4] = {0,0,0,0}`

- `u_int64_t totalpeaksearch [3] = {0,0,0}`

- `u_int64_t totalpeakenclosing = 0`

*This variable counts the total number of peak enclosings. One peak search contains several enclosings.*

- `u_int64_t totalrescheduled = {0}`

*This variable counts the total number of re-scheduling.*

- `u_int64_t totalfiltered_maxgrad = 0`

*This variable counts the total filtering caused by maximum gradient check.*

- `u_int64_t totalfiltered_incontinuity = 0`

*This variable counts the total filtering caused by nearby incontinuity.*

- `u_int64_t totalfiltered_nextpulse = 0`

*This variable counts the total filtering caused by nearby next pulse.*

### 8.2.1 Variable Documentation

#### 8.2.1.1 `u_int64_t punnets_common::totalpartition_newton = {0,0,0,0}`

This variable counts the number of partitions with Newton-Raphson search. Each element contains the number of 0th, 1st, 2nd. and delta partitions, respectively.

Definition at line 42 of file `dneuron.cpp`.

Referenced by `punnets_private::tneuron_ext< debug >::scheduleFire()`.

#### 8.2.1.2 `u_int64_t punnets_common::totalpartition_nonewton = {0,0,0,0}`

This variable counts the number of partitions without Newton-Raphson search. Each element contains the number of 0th, 1st, 2nd. and delta partitions, respectively.

Definition at line 41 of file `dneuron.cpp`.

Referenced by `punnets_private::tneuron_ext< debug >::scheduleFire()`.

**8.2.1.3** `u_int64_t punnets_common::totalpeaksearch = {0,0,0}`

This variable counts the number of peak searches. Each element contains the number of searches with peak below threshold (no crossing), peak above threshold (crossing), no peak (not convex), respectively.

Definition at line 43 of file `dneuron.cpp`.

Referenced by `punnets_private::tneuron_ext< debug >::scheduleFire()`.

## 8.3 Neurons

### Compounds

- class `tneuron`
- class `tneuron_base`
- class `tneuron_ext`
- class `tneuron_ext_const`



## 8.4 Synapses

### Compounds

- class `tsynapse`
- class `tsynapse_addfunc`

*Synapse class that adds a new function to the destination (postsynaptic) `tneuron_ext` (p. 81).*

- class `tsynapse_base`
- class `tsynapse_fatigue`
- class `tsynapse_message`
- class `tsynapse_messfunc`

## 8.5 Scheduling

### Compounds

- struct `greater_tevent`
- class `taction`
- class `tevent`
- class `tsched_double`

# Chapter 9

## Punnets Namespace Documentation

### 9.1 punnets Namespace Reference

#### Typedefs

- typedef punnets\_private::tsynapse< true > **tsynapse**  
*a typedef referring punnets\_private::tsynapse (p. 90) class.*
- typedef punnets\_private::tsynapse\_message< true > **tsynapse\_message**  
*a typedef referring punnets\_private::tsynapse\_message (p. 98) class.*
- typedef punnets\_private::tsynapse\_fatigue< true > **tsynapse\_fatigue**  
*a typedef referring punnets\_private::tsynapse\_fatigue (p. 96) class.*
- typedef punnets\_private::tsynapse\_addfunc< true > **tsynapse\_addfunc**  
*a typedef referring punnets\_private::tsynapse\_addfunc (p. 92) class.*
- typedef punnets\_private::tsynapse\_messfunc< true > **tsynapse\_messfunc**  
*a typedef referring punnets\_private::tsynapse\_messfunc (p. 100) class.*
- typedef punnets\_private::tneuron< true > **tneuron**  
*a typedef referring punnets\_private::tneuron (p. 76) class.*
- typedef punnets\_private::tneuron\_ext\_const< true > **tneuron\_ext\_const**  
*a typedef referring punnets\_private::tneuron\_ext\_const (p. 84) class.*
- typedef punnets\_private::tneuron\_ext< true > **tneuron\_ext**  
*a typedef referring punnets\_private::tneuron\_ext (p. 81) class.*

### 9.1.1 Detailed Description

This namespace provides Punnets library. Importing this namespace, a user can access the classes of Punnets library. Alternatively, if you need efficiency, you may import **punnets\_nodebug** (p. 29) namespace. For most of the available classes, refer to the **punnets\_common** (p. 23) namespace.

This namespace imports **punnets\_common** (p. 23) so that every definitions in the **punnets\_common** (p. 23) namespace can be transparently used in this namespace.

## 9.2 punnets\_common Namespace Reference

This namespace provides commonly available classes. Users should use classes and functions in this namespace via either punnets or **punnets\_nodebug** (p. 29) namespace.

### Compounds

- class **debugflag**< false >
- class **debugflag**< true >

*This instantiation of the class is used for punnets namespace.*

- class **func\_base**
- class **func\_const**

</classdef>

- class **func\_const\_int**
- class **func\_delta\_int**

</classdef>

- class **func\_deriveq\_base**
- class **func\_exp**
- class **func\_exp\_diff**
- class **func\_exp\_int**
- class **func\_response**
- class **func\_sine**
- class **func\_sine\_int**
- class **func\_sineshot**
- class **func\_sineshot\_int**
- class **func\_step**

</classdef>

- struct **greater\_tevent**
- class **message\_add\_pulse**

*A message that adds a new pulse.*

- class **message\_add\_event\_time**
- class **message\_base**
- class **message\_set\_lambda**

*A message that changes lambda (leak value).*

- class **message\_set\_zero\_point**

*A message that changes zero point.*

- class **message\_set\_t0**

*A message that changes t0.*

- struct **less\_sched\_entry**

*This class is used to construct a global priority queue on STL.*

- class **taction**

- class **tevent**
- struct **neumentry**  
*A class to keep a list of logging neurons.*

- class **tlogger**
- class **tmessage**
- class **tneuron\_base**
- class **tpulse**
- class **tsched\_double**
- class **tsentinel**
- class **tsynapse\_base**

## Typedefs

- typedef double **real**
- typedef double **ntime\_t**

## Functions

- **taction** & **makePulse** (**tneuron\_base** &idest, **real** ilevel)  
*Generate a pulse to the specified destination.*
- **taction** & **makePulse** (**tneuron\_base** &idest, **message\_base** \*mess)  
*Generate a pulse to the specified destination.*
- template<bool b> **taction** & **setExtInput** (**punnets\_private::tneuron\_ext\_const**< b > &idest, **real** ilevel)  
*Set an external input to a specified level.*

## Variables

- **u\_int64\_t totalfire** = 0  
*This variable counts the total number of firing of the neurons.*
- **u\_int64\_t totalpulse** = 0  
*This variable counts the total pulse arrivals to any of the neurons.*
- **u\_int64\_t totalpartition** = 0  
*This variable counts the total number of partitions.*
- **u\_int64\_t totalpartition\_nonewton** [4] = {0,0,0,0}
- **u\_int64\_t totalpartition\_newton** [4] = {0,0,0,0}
- **u\_int64\_t totalpeaksearch** [3] = {0,0,0}
- **u\_int64\_t totalpeakenclosing** = 0  
*This variable counts the total number of peak enclosings. One peak search contains several enclosings.*
- **u\_int64\_t totalrescheduled** = {0}

*This variable counts the total number of re-scheduling.*

- `u_int64_t totalfiltered_maxgrad = 0`

*This variable counts the total filtering caused by maximum gradient check.*

- `u_int64_t totalfiltered_incontinuity = 0`

*This variable counts the total filtering caused by nearby incontinuity.*

- `u_int64_t totalfiltered_nextpulse = 0`

*This variable counts the total filtering caused by nearby next pulse.*

- `const real epsilon = 1e-10`

*This variable specifies the minimum error value.*

## 9.2.1 Detailed Description

This namespace provides commonly available classes. Users should use classes and functions in this namespace via either `punnets` or `punnets_nodebug` (p. 29) namespace.

## 9.2.2 Typedef Documentation

### 9.2.2.1 typedef double punnets\_common::ntime\_t

Time representation type.

The type that represents a simulation time.

Definition at line 34 of file `punnets_base.h`.

Referenced by `punnets_common::tsentinel::activate()`, `punnets_private::tsynapse_messfunc< debug >::activate()`, `punnets_private::tsynapse_addfunc< debug >::activate()`, `punnets_private::tneuron_ext< debug >::activate()`, `punnets_private::tneuron< debug >::activate()`, `punnets_private::tsynapse_fatigue< debug >::activate()`, `punnets_private::tsynapse_message< debug >::activate()`, `punnets_private::tsynapse< debug >::activate()`, `punnets_common::tlogger::activate()`, `punnets_common::tlogger::add()`, `punnets_private::tsynapse_fatigue< debug >::addDelay()`, `punnets_private::tsynapse< debug >::addDelay()`, `punnets_private::tneuron_ext< debug >::broadcastMessage()`, `punnets_private::tneuron_ext< debug >::calcSignal()`, `punnets_private::tneuron_ext< debug >::fire()`, `punnets_common::func_delta_int::func_delta_int()`, `punnets_common::func_exp_diff::get1stDeriv()`, `punnets_common::func_exp_int::get1stDeriv()`, `punnets_common::func_exp::get1stDeriv()`, `punnets_common::func_sineshot_int::get1stDeriv()`, `punnets_common::func_sine_int::get1stDeriv()`, `punnets_common::func_sineshot::get1stDeriv()`, `punnets_common::func_sine::get1stDeriv()`, `punnets_common::func_response::get1stDeriv()`, `punnets_common::func_delta_int::get1stDeriv()`, `punnets_common::func_step::get1stDeriv()`, `punnets_common::func_const_int::get1stDeriv()`, `punnets_common::func_const::get1stDeriv()`, `punnets_common::func_exp_int::get1stDerivDomain()`, `punnets_common::func_exp::get1stDerivDomain()`, `punnets_common::func_sine_int::get1stDerivDomain()`, `punnets_common::func_step::get1stDerivDomain()`, `punnets_common::func_const::get1stDerivDomain()`, `punnets_common::func_exp_diff::get1stDerivDomain()`, `punnets_common::func_sineshot_int::get1stDerivDomain()`, `punnets_common::func_sineshot::get1stDerivDomain()`, `punnets_common::func_sine::get1stDerivDomain()`, `punnets_common::func_response::get1stDerivDomain()`, `punnets_common::func_delta_int::get1stDerivDomain()`, `punnets_common::func_const_int::get1stDerivDomain()`, `punnets_common::func_exp_diff::get2nd`

Deriv(), punnets\_common::func\_exp\_int::get2ndDeriv(), punnets\_common::func\_exp::get2ndDeriv(), punnets\_common::func\_sineshot\_int::get2ndDeriv(), punnets\_common::func\_sine\_int::get2ndDeriv(), punnets\_common::func\_sineshot::get2ndDeriv(), punnets\_common::func\_sine::get2ndDeriv(), punnets\_common::func\_response::get2ndDeriv(), punnets\_common::func\_delta\_int::get2ndDeriv(), punnets\_common::func\_step::get2ndDeriv(), punnets\_common::func\_const\_int::get2ndDeriv(), punnets\_common::func\_const::get2ndDeriv(), punnets\_common::func\_exp\_int::get2ndDerivDomain(), punnets\_common::func\_exp::get2ndDerivDomain(), punnets\_common::func\_sine\_int::get2ndDerivDomain(), punnets\_common::func\_step::get2ndDerivDomain(), punnets\_common::func\_const::get2ndDerivDomain(), punnets\_common::func\_exp\_diff::get2ndDerivDomain(), punnets\_common::func\_sineshot\_int::get2ndDerivDomain(), punnets\_common::func\_sineshot::get2ndDerivDomain(), punnets\_common::func\_sine::get2ndDerivDomain(), punnets\_common::func\_response::get2ndDerivDomain(), punnets\_common::func\_delta\_int::get2ndDerivDomain(), punnets\_common::func\_const\_int::get2ndDerivDomain(), punnets\_private::tneuron\_ext\_const< debug >::getCurrentExtInput(), punnets\_common::tneuron\_base::getCurrentExtInput(), punnets\_private::tneuron\_ext< debug >::getCurrentSigLevel(), punnets\_private::tneuron\_ext\_const< debug >::getCurrentSigLevel(), punnets\_private::tneuron< debug >::getCurrentSigLevel(), punnets\_common::tneuron\_base::getCurrentSigLevel(), punnets\_private::tneuron\_ext< debug >::getCurrentThrLevel(), punnets\_private::tneuron< debug >::getCurrentThrLevel(), punnets\_common::tneuron\_base::getCurrentThrLevel(), punnets\_private::tneuron\_ext< debug >::getCurrentThrLevel(), punnets\_common::tsynapse\_base::getDelay(), punnets\_private::tneuron\_ext< debug >::getLastFire(), punnets\_common::tneuron\_base::getLastFire(), punnets\_private::tneuron\_ext< debug >::getLastSimulate(), punnets\_private::tneuron< debug >::getLastSimulate(), punnets\_common::tneuron\_base::getLastSimulate(), punnets\_common::func\_exp\_diff::getMaxGradient(), punnets\_common::func\_exp\_int::getMaxGradient(), punnets\_common::func\_exp::getMaxGradient(), punnets\_common::func\_sineshot\_int::getMaxGradient(), punnets\_common::func\_sine\_int::getMaxGradient(), punnets\_common::func\_sineshot::getMaxGradient(), punnets\_common::func\_sine::getMaxGradient(), punnets\_common::func\_response::getMaxGradient(), punnets\_common::func\_delta\_int::getMaxGradient(), punnets\_common::func\_step::getMaxGradient(), punnets\_common::func\_const\_int::getMaxGradient(), punnets\_common::func\_const::getMaxGradient(), punnets\_common::func\_exp\_diff::getNextIncontinuity(), punnets\_common::func\_exp\_int::getNextIncontinuity(), punnets\_common::func\_exp::getNextIncontinuity(), punnets\_common::func\_sineshot::getNextIncontinuity(), punnets\_common::func\_sine::getNextIncontinuity(), punnets\_common::func\_response::getNextIncontinuity(), punnets\_common::func\_delta\_int::getNextIncontinuity(), punnets\_common::func\_step::getNextIncontinuity(), punnets\_common::func\_base::getNextIncontinuity(), punnets\_common::tevent::getTime(), punnets\_common::func\_exp\_diff::getValue(), punnets\_common::func\_exp\_int::getValue(), punnets\_common::func\_exp::getValue(), punnets\_common::func\_sineshot\_int::getValue(), punnets\_common::func\_sine\_int::getValue(), punnets\_common::func\_sineshot::getValue(), punnets\_common::func\_sine::getValue(), punnets\_common::func\_response::getValue(), punnets\_common::func\_delta\_int::getValue(), punnets\_common::func\_step::getValue(), punnets\_common::func\_const\_int::getValue(), punnets\_common::func\_const::getValue(), punnets\_common::func\_exp\_int::getValueDomain(), punnets\_common::func\_exp::getValueDomain(), punnets\_common::func\_sine\_int::getValueDomain(), punnets\_common::func\_step::getValueDomain(), punnets\_common::func\_const::getValueDomain(), punnets\_common::func\_exp\_diff::getValueDomain(), punnets\_common::func\_sineshot\_int::getValueDomain(), punnets\_common::func\_sineshot::getValueDomain(), punnets\_common::func\_sine::getValueDomain(), punnets\_common::func\_response::getValueDomain(), punnets\_common::func\_delta\_int::getValueDomain(), punnets\_common::func\_const\_int::getValueDomain(), punnets\_common::func\_sineshot::processMessage(), punnets\_common::func\_delta\_int::processMessage(), punnets\_common::func\_deriveq\_base::processMessage(), punnets\_common::func\_base::processMessage(), punnets\_common::func\_exp\_diff::processMessage(), tobserver::pulseArrive(), punnets\_private::tneuron< debug >::pulseArrive(), punnets\_private::tneuron\_ext< debug >::pulseArrive(), punnets\_common::tsched\_double::run(), punnets\_common::tsched\_double::scheduleEvent(), punnets\_private::tneuron\_ext< debug >::scheduleFire(), pun-



nets\_private::tneuron\_ext\_const< debug >::scheduleFire(), punnets\_private::tneuron< debug >::scheduleFire(), punnets\_private::tneuron\_ext< debug >::sendMessage(), punnets\_private::tneuron\_ext\_const< debug >::setConvergeLevel(), punnets\_private::tneuron\_ext\_const< debug >::setExtInput(), punnets\_private::tneuron\_ext< debug >::setLoopBack(), punnets\_common::func\_sineshot::shouldDelete(), punnets\_common::func\_base::shouldDelete(), punnets\_private::tneuron\_ext\_const< debug >::simulateElastic(), punnets\_private::tneuron< debug >::simulateElastic(), punnets\_common::tevent::tevent(), punnets\_common::tlogger::tlogger(), punnets\_private::tneuron< debug >::neuron(), punnets\_private::tneuron\_ext\_const< debug >::tneuron\_ext\_const(), punnets\_private::tsynapse< debug >::tsynapse(), and punnets\_common::tsynapse\_base::tsynapse\_base().

### 9.2.2.2 typedef double punnets\_common::real

Real number type.

The type that represents a real number.

Definition at line 28 of file punnets\_base.h.

Referenced by punnets\_private::tsynapse\_fatigue< debug >::activate(), punnets\_common::tlogger::add(), punnets\_private::tsynapse\_fatigue< debug >::addWeight(), punnets\_private::tsynapse< debug >::addWeight(), punnets\_private::tneuron\_ext< debug >::calcSignal(), punnets\_common::func\_const::func\_const(), punnets\_common::func\_const\_int::func\_const\_int(), punnets\_common::func\_delta\_int::func\_delta\_int(), punnets\_common::func\_exp::func\_exp(), punnets\_common::func\_exp\_diff::func\_exp\_diff(), punnets\_common::func\_exp\_int::func\_exp\_int(), punnets\_common::func\_sine::func\_sine(), punnets\_common::func\_sine\_int::func\_sine\_int(), punnets\_common::func\_sineshot::func\_sineshot(), punnets\_common::func\_sineshot\_int::func\_sineshot\_int(), punnets\_common::func\_exp\_diff::get1stDeriv(), punnets\_common::func\_exp\_int::get1stDeriv(), punnets\_common::func\_exp::get1stDeriv(), punnets\_common::func\_sineshot\_int::get1stDeriv(), punnets\_common::func\_sine\_int::get1stDeriv(), punnets\_common::func\_sineshot::get1stDeriv(), punnets\_common::func\_sine::get1stDeriv(), punnets\_common::func\_response::get1stDeriv(), punnets\_common::func\_delta\_int::get1stDeriv(), punnets\_common::func\_step::get1stDeriv(), punnets\_common::func\_const\_int::get1stDeriv(), punnets\_common::func\_const::get1stDeriv(), punnets\_common::func\_exp\_int::get1stDerivDomain(), punnets\_common::func\_exp::get1stDerivDomain(), punnets\_common::func\_sine\_int::get1stDerivDomain(), punnets\_common::func\_step::get1stDerivDomain(), punnets\_common::func\_const::get1stDerivDomain(), punnets\_common::func\_exp\_diff::get1stDerivDomain(), punnets\_common::func\_sineshot\_int::get1stDerivDomain(), punnets\_common::func\_sineshot::get1stDerivDomain(), punnets\_common::func\_sine::get1stDerivDomain(), punnets\_common::func\_response::get1stDerivDomain(), punnets\_common::func\_delta\_int::get1stDerivDomain(), punnets\_common::func\_const\_int::get1stDerivDomain(), punnets\_common::func\_exp\_diff::get2ndDeriv(), punnets\_common::func\_exp\_int::get2ndDeriv(), punnets\_common::func\_exp::get2ndDeriv(), punnets\_common::func\_sineshot\_int::get2ndDeriv(), punnets\_common::func\_sine\_int::get2ndDeriv(), punnets\_common::func\_sineshot::get2ndDeriv(), punnets\_common::func\_sine::get2ndDeriv(), punnets\_common::func\_response::get2ndDeriv(), punnets\_common::func\_delta\_int::get2ndDeriv(), punnets\_common::func\_step::get2ndDeriv(), punnets\_common::func\_const\_int::get2ndDeriv(), punnets\_common::func\_const::get2ndDeriv(), punnets\_common::func\_exp\_int::get2ndDerivDomain(), punnets\_common::func\_exp::get2ndDerivDomain(), punnets\_common::func\_sine\_int::get2ndDerivDomain(), punnets\_common::func\_step::get2ndDerivDomain(), punnets\_common::func\_const::get2ndDerivDomain(), punnets\_common::func\_exp\_diff::get2ndDerivDomain(), punnets\_common::func\_sineshot\_int::get2ndDerivDomain(), punnets\_common::func\_sineshot::get2ndDerivDomain(), punnets\_common::func\_sine::get2ndDerivDomain(), punnets\_common::func\_response::get2ndDerivDomain(), punnets\_common::func\_delta\_int::get2ndDerivDomain(), punnets\_common::func\_const\_int::get2ndDerivDomain(), punnets\_private::tneuron\_ext\_const< debug >::getConvergeLevel(), punnets\_private::tneuron\_ext\_const<

`debug >::getCurrentExtInput()`, `punnets_common::tneuron_base::getCurrentExtInput()`, `punnets_private::tneuron_ext< debug >::getCurrentSigLevel()`, `punnets_private::tneuron_ext_const< debug >::getCurrentSigLevel()`, `punnets_private::tneuron< debug >::getCurrentSigLevel()`, `punnets_common::tneuron_base::getCurrentSigLevel()`, `punnets_private::tneuron_ext< debug >::getCurrentThrLevel()`, `punnets_private::tneuron< debug >::getCurrentThrLevel()`, `punnets_common::tneuron_base::getCurrentThrLevel()`, `punnets_private::tneuron_ext_const< debug >::getExtInput()`, `punnets_common::func_exp_diff::getMaxGradient()`, `punnets_common::func_exp_int::getMaxGradient()`, `punnets_common::func_exp::getMaxGradient()`, `punnets_common::func_sineshot_int::getMaxGradient()`, `punnets_common::func_sine_int::getMaxGradient()`, `punnets_common::func_sineshot::getMaxGradient()`, `punnets_common::func_sine::getMaxGradient()`, `punnets_common::func_response::getMaxGradient()`, `punnets_common::func_delta_int::getMaxGradient()`, `punnets_common::func_step::getMaxGradient()`, `punnets_common::func_const_int::getMaxGradient()`, `punnets_common::func_const::getMaxGradient()`, `punnets_common::func_exp_diff::getValue()`, `punnets_common::func_exp_int::getValue()`, `punnets_common::func_exp::getValue()`, `punnets_common::func_sineshot_int::getValue()`, `punnets_common::func_sine_int::getValue()`, `punnets_common::func_sineshot::getValue()`, `punnets_common::func_sine::getValue()`, `punnets_common::func_response::getValue()`, `punnets_common::func_delta_int::getValue()`, `punnets_common::func_step::getValue()`, `punnets_common::func_const_int::getValue()`, `punnets_common::func_const::getValue()`, `punnets_common::func_exp_int::getValueDomain()`, `punnets_common::func_exp::getValueDomain()`, `punnets_common::func_sine_int::getValueDomain()`, `punnets_common::func_step::getValueDomain()`, `punnets_common::func_const::getValueDomain()`, `punnets_common::func_exp_diff::getValueDomain()`, `punnets_common::func_sineshot_int::getValueDomain()`, `punnets_common::func_sineshot::getValueDomain()`, `punnets_common::func_sine::getValueDomain()`, `punnets_common::func_response::getValueDomain()`, `punnets_common::func_delta_int::getValueDomain()`, `punnets_common::func_const_int::getValueDomain()`, `punnets_private::tsynapse_fatigue< debug >::getWeight()`, `punnets_private::tsynapse< debug >::getWeight()`, `punnets_common::tsynapse_base::getWeight()`, `makePulse()`, `tobserver::pulseArrive()`, `punnets_private::tneuron< debug >::pulseArrive()`, `punnets_private::tneuron_ext< debug >::pulseArrive()`, `punnets_private::tneuron_ext< debug >::scheduleFire()`, `punnets_private::tneuron_ext_const< debug >::scheduleFire()`, `punnets_private::tneuron< debug >::scheduleFire()`, `punnets_private::tneuron_ext_const< debug >::setConvergeLevel()`, `punnets_private::tneuron_ext< debug >::setExtInput()`, `punnets_common::func_deriveq_base::setLambda()`, `punnets_common::func_base::setLambda()`, `punnets_common::func_response::setZeroPoint()`, `punnets_common::func_deriveq_base::setZeroPoint()`, `punnets_common::func_base::setZeroPoint()`, `punnets_private::tneuron< debug >::tneuron()`, `punnets_private::tneuron_ext< debug >::tneuron_ext()`, `punnets_private::tneuron_ext_const< debug >::tneuron_ext_const()`, `punnets_private::tsynapse< debug >::tsynapse()`, `punnets_private::tsynapse_addfunc< debug >::tsynapse_addfunc()`, `punnets_private::tsynapse_fatigue< debug >::tsynapse_fatigue()`, `punnets_private::tsynapse_message< debug >::tsynapse_message()`, and `punnets_private::tsynapse_messfunc< debug >::tsynapse_messfunc()`.

## 9.3 punnets\_nodebug Namespace Reference

### Typedefs

- typedef punnets\_private::tsynapse< false > **tsynapse**  
*tsynapse class.*
- typedef punnets\_private::tsynapse\_message< false > **tsynapse\_message**  
*tsynapse\_message class.*
- typedef punnets\_private::tsynapse\_fatigue< false > **tsynapse\_fatigue**  
*tsynapse\_fatigue class.*
- typedef punnets\_private::tsynapse\_addfunc< false > **tsynapse\_addfunc**  
*tsynapse\_addfunc class.*
- typedef punnets\_private::tsynapse\_messfunc< false > **tsynapse\_messfunc**  
*tsynapse\_messfunc class.*
- typedef punnets\_private::tneuron< false > **tneuron**  
*tneuron class.*
- typedef punnets\_private::tneuron\_ext\_const< false > **tneuron\_ext\_const**  
*tneuron\_ext\_const class.*
- typedef punnets\_private::tneuron\_ext< false > **tneuron\_ext**  
*tneuron\_ext class.*

### 9.3.1 Detailed Description

This namespace provides Punnets library without debugging facility. Importing this namespace, a user can access an efficient variant of the Punnets library. It is recommended for a user to use **punnets** (p. 21) namespace, at least in the period of development. For most of the available classes, refer to the **punnets\_common** (p. 23) namespace.

This namespace imports **punnets\_common** (p. 23) so that every definitions in the **punnets-common** (p. 23) namespace can be transparently used in this namespace.

## 9.4 punnets\_private Namespace Reference

This namespace provides some private classes used for punnets library. Users should not use classes and functions in this namespace directly.

### Compounds

- class **tneuron**
- class **tneuron\_ext**
- class **tneuron\_ext\_const**
- class **tsynapse**
- class **tsynapse\_addfunc**

*Synapse class that adds a new function to the destination (postsynaptic) **tneuron\_ext** (p. 81).*

- class **tsynapse\_fatigue**
- class **tsynapse\_message**
- class **tsynapse\_messfunc**
- struct **valdomain**

### 9.4.1 Detailed Description

This namespace provides some private classes used for punnets library. Users should not use classes and functions in this namespace directly.

# Chapter 10

## Punnets Class Documentation

### 10.1 punnets\_common::debugflag< false > Class Template Reference

```
#include <dneuron.h>
```

#### Public Methods

- `bool getDeb () const`  
*Get debugging condition, which is always false.*
- `void setDeb (bool)`  
*Set debugging condition, which always fails.*

#### 10.1.1 Detailed Description

```
template<> class punnets_common::debugflag< false >
```

This instantiation of the class is used for `punnets_nodebug` (p. 29) namespace. it provides always false value to the `getDeb()` (p. 31), so that the debugging code is optimized out.

Definition at line 114 of file `dneuron.h`.

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

- `dneuron.h`

## 10.2 punnets\_common::debugflag< true > Class Template Reference

This instantiation of the class is used for punnets namespace.

```
#include <dneuron.h>
```

### Public Methods

- **debugflag** ()  
*Constructor initialized the debugging condition to false.*
- **bool getDeb** () const  
*Get debugging condition.*
- **void setDeb** (bool b)  
*Set debugging condition.*

### Protected Attributes

- **bool debug**  
*The debugging condition.*

#### 10.2.1 Detailed Description

```
template<> class punnets_common::debugflag< true >
```

This instantiation of the class is used for punnets namespace.

Definition at line 125 of file dneuron.h.

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

- **dneuron.h**

## 10.3 punnets\_common::func\_base Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_base::



### Public Methods

- virtual bool **shouldDelete** (**ntime\_t**)  
Return true if the function will return only zeros after the specified time.
- virtual bool **processMessage** (**ntime\_t**, const **message\_base** &)
- virtual void **setLambda** (**real**)  
Change leak value on a leaky integrate function. Redefined in **func\_deriveq\_base** (p. 43).
- virtual void **setZeroPoint** (**real**)  
Change zero point on a leaky integrate function. Redefined in **func\_deriveq\_base** (p. 43).
- virtual **real** **getMaxGradient** (**ntime\_t** t) const=0  
A pure virtual function that returns the max gradient of the function after the time t.
- virtual **ntime\_t** **getNextIncontinuity** (**ntime\_t**) const
- virtual **real** **getValue** (**ntime\_t** t) const=0  
A pure virtual function that returns  $f(t)$ .
- virtual **real** **get1stDeriv** (**ntime\_t** t) const=0  
A pure virtual function that returns  $df(t)/dt$ .
- virtual **real** **get2ndDeriv** (**ntime\_t** t) const=0  
A pure virtual function that returns  $d^2f(t)/dt^2$ .
- virtual void **getValueDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const=0  
Obtains a zeroth-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .
- virtual void **get1stDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const=0  
Obtains a first-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .
- virtual void **get2ndDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const=0  
Obtains a second-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .
- virtual **func\_base** \* **clone** ()=0

*Allocates and returns another instance of this function.*

- virtual std::string **getDescription** ()=0  
*Obtains a human-readable description string of this function.*

## Protected Methods

- virtual void **valueChange** ()

### 10.3.1 Detailed Description

Abstract base class of a function.

A base class that represents a numerical function. Some functions needs a reference point to be calculated based on differential equation. Such a function is derived from **func\_deriveq\_base** (p. 43), but **func\_base** (p. 33) contains interface for such a function.

Definition at line 69 of file func.h.

### 10.3.2 Member Function Documentation

#### 10.3.2.1 virtual ntime\_t punnets\_common::func\_base::getNextIncontinuity (ntime\_t) const [inline, virtual]

A virtual function that returns the next incontinuity point after the time t. In **func\_base** (p. 33), the function always returns infinity.

Reimplemented in **punnets\_common::func\_step** (p. 66), **punnets\_common::func\_delta\_int** (p. 41), **punnets\_common::func\_response** (p. 54), **punnets\_common::func\_sine** (p. 56), **punnets\_common::func\_sinshot** (p. 60), **punnets\_common::func\_exp** (p. 47), **punnets\_common::func\_exp\_int** (p. 52), and **punnets\_common::func\_exp\_diff** (p. 49).

Definition at line 96 of file func.h.

References punnets\_common::ntime\_t.

```
96 { return mak::Infinity; };
```

#### 10.3.2.2 virtual bool punnets\_common::func\_base::processMessage (ntime\_t, const message\_base &) [inline, virtual]

Process the specified message at the specified time. Return true if the message is processed. In **func\_base** (p. 33), this function always returns false (processes no message).

Reimplemented in **punnets\_common::func\_deriveq\_base** (p. 43), **punnets\_common::func\_delta\_int** (p. 40), **punnets\_common::func\_sinshot** (p. 61), and **punnets\_common::func\_exp\_diff** (p. 49).

Definition at line 85 of file func.h.

References punnets\_common::ntime\_t.

```
85 { return false; }
```



### 10.3.2.3 virtual void punnets\_common::func\_base::valueChange () [inline, protected, virtual]

An entrance to recalculate coefficient at the value change caused by some external factor. In **func\_base** (p.33) this function do nothing. This function will be defined in the derived classes with the requirement of the recalculation.

Definition at line 75 of file func.h.

Referenced by punnets\_common::func\_const\_int::func\_const\_int(), punnets\_common::func\_delta\_int::func\_delta\_int(), punnets\_common::func\_exp\_diff::func\_exp\_diff(), punnets\_common::func\_exp\_int::func\_exp\_int(), punnets\_common::func\_sine::func\_sine(), punnets\_common::func\_sine\_int::func\_sine\_int(), punnets\_common::func\_sineshot::func\_sineshot(), punnets\_common::func\_sineshot\_int::func\_sineshot\_int(), punnets\_common::func\_exp\_diff::processMessage(), and punnets\_common::func\_deriveq\_base::setLambda().

```
75 { }
```

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

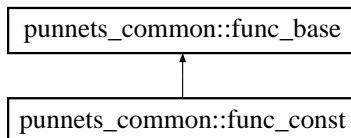
- **func.h**

## 10.4 punnets\_common::func\_const Class Reference

</classdef>

#include <func.h>

Inheritance diagram for punnets\_common::func\_const::



### Public Methods

- **func\_const** (real ic)  
*constructs a **func\_const** (p.36) with the constant c.*
- virtual **real getMaxGradient** (ntime\_t) const  
*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (ntime\_t) const  
*A pure virtual function that returns f(t).*
- virtual **real get1stDeriv** (ntime\_t) const  
*A pure virtual function that returns df(t)/dt.*
- virtual **real get2ndDeriv** (ntime\_t) const  
*A pure virtual function that returns d<sup>2</sup>f(t)/dt<sup>2</sup>.*
- virtual void **getValueDomain** (ntime\_t, real &upslope, real &ceil, real &downslope, real &floor) const  
*Obtains a zeroth-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual void **get1stDerivDomain** (ntime\_t, real &upslope, real &ceil, real &downslope, real &floor) const  
*Obtains a first-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual void **get2ndDerivDomain** (ntime\_t, real &upslope, real &ceil, real &downslope, real &floor) const  
*Obtains a second-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **func\_const \* clone** ()  
*Allocates and returns another instance of this function.*
- virtual std::string **getDescription** ()  
*Obtains a human-readable description string of this function.*

### 10.4.1 Detailed Description

</classdef>

A constant function.

This class represents a constant function  $f(t) = c$ .

Definition at line 201 of file func.h.

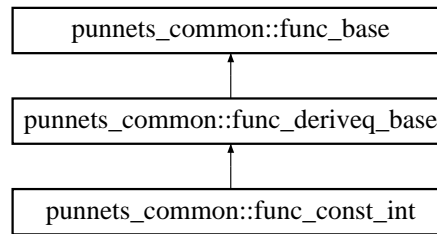
The documentation for this class was generated from the following file:

- **func.h**

## 10.5 punnets\_common::func\_const\_int Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_const\_int::



### Public Methods

- **func\_const\_int** (real ic)
 

*Constructs a **func\_const\_int** (p.38) with the constant c.*
- virtual **real getMaxGradient** (ntime\_t t) const
 

*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (ntime\_t t) const
 

*A pure virtual function that returns  $f(t)$ .*
- virtual **real get1stDeriv** (ntime\_t t) const
 

*A pure virtual function that returns  $df(t)/dt$ .*
- virtual **real get2ndDeriv** (ntime\_t t) const
 

*A pure virtual function that returns  $d^2f(t)/dt^2$ .*
- virtual void **getValueDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a zeroth-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get1stDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a first-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get2ndDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a second-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual **func\_const\_int \* clone** ()
 

*Allocates and returns another instance of this function.*
- virtual std::string **getDescription** ()

*Obtains a human-readable description string of this function.*

### 10.5.1 Detailed Description

An integration of a constant function.

The result of solving  $dx/dt = c - \lambda x$ . That is,  $x = c/\lambda + C e^{-\lambda t}$ , where  $C$  is an integration constant.

Definition at line 238 of file func.h.

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

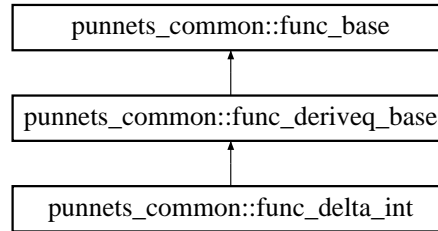
- **func.h**
- **func.cpp**

## 10.6 punnets\_common::func\_delta\_int Class Reference

</classdef>

#include <func.h>

Inheritance diagram for punnets\_common::func\_delta\_int::



### Public Methods

- **func\_delta\_int** (**real** ir, **ntime\_t** it0)
 

*Constructs **func\_delta\_int** (p. 40) with pulse amplitude r, and pulse arrival time t0.*
- virtual **real** **getMaxGradient** (**ntime\_t** t) const
 

*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real** **getValue** (**ntime\_t** t) const
- virtual **real** **get1stDeriv** (**ntime\_t** t) const
 

*A pure virtual function that returns  $df(t)/dt$ .*
- virtual **real** **get2ndDeriv** (**ntime\_t** t) const
 

*A pure virtual function that returns  $d^2f(t)/dt^2$ .*
- virtual **ntime\_t** **getNextIncontinuity** (**ntime\_t** from) const
- virtual void **getValueDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const
 

*Obtains a zeroth-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get1stDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const
 

*Obtains a first-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get2ndDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const
 

*Obtains a second-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual std::string **getDescription** ()
 

*Obtains a human-readable description string of this function.*
- virtual bool **processMessage** (**ntime\_t** t, const **message\_base** &m)

*Processes message\_set\_lambda and message\_set\_zero\_point messages.*

- virtual `func_delta_int * clone ()`

*Allocates and returns another instance of this function.*

### 10.6.1 Detailed Description

</classdef>

An integrated delta function with leak.

A leaky integration of a delta function.  $\$df/dt = r \backslash\delta( t - t_0 ) - \backslash\lambda f\$$ .

Definition at line 316 of file func.h.

### 10.6.2 Member Function Documentation

#### 10.6.2.1 virtual `ntime_t punnets_common::func_delta_int::getNextIncontinuity (ntime_t from) const` [inline, virtual]

A virtual function that returns the next incontinuity point after the time  $t$ . In `func_base` (p. 33), the function always returns infinity.

Reimplemented from `punnets_common::func_base` (p. 34).

Definition at line 345 of file func.h.

References `punnets_common::ntime_t`.

```
345 { return from < t0 ? t0 : mak::Infinity; };
```

#### 10.6.2.2 virtual `real punnets_common::func_delta_int::getValue (ntime_t t) const` [inline, virtual]

A function that returns a leaky integrated value between the last pulse arrival time  $t_0$  and  $t$ .  $\$x = 1/e^{\&\lambda t} ( \&\int_{from}^{t_0} e^{\&\lambda t} f(t) + C )\$$

Implements `punnets_common::func_base` (p. 33).

Definition at line 331 of file func.h.

References `punnets_common::ntime_t`, and `punnets_common::real`.

Referenced by `processMessage()`.

```
334         {
335 //             std::cout << "zerop=" << zerop << ", t0=" << t0 << ",t=" << t << std::endl;
336             return (zerop < t0 && t0 <= t) ? r * exp( - lambda * ( t - t0 ) ) : 0.0; }
```

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

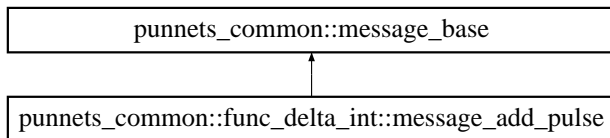
- `func.h`
- `func.cpp`

## 10.7 punnets\_common::func\_delta\_int::message\_add\_pulse Class Reference

A message that adds a new pulse.

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_delta\_int::message\_add\_pulse:



### Public Methods

- virtual const char \* **getMessageId** () const

#### 10.7.1 Detailed Description

A message that adds a new pulse.

Definition at line 354 of file func.h.

#### 10.7.2 Member Function Documentation

##### 10.7.2.1 virtual const char\* punnets\_common::func\_delta\_int::message\_add\_pulse::getMessageId () [inline, virtual]

Returns the pointer of a string that represents the class. The pointer must be same for any instance of a given class. A function can check the type of the message by comparing the pointer.

Implements **punnets\_common::message\_base** (p. 68).

Definition at line 360 of file func.h.

```
360 { return messageId; }
```

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

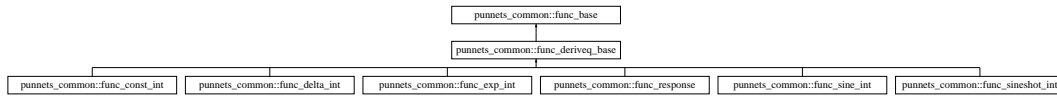
- **func.h**
- **func.cpp**



## 10.8 punnets\_common::func\_deriveq\_base Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_deriveq\_base::



### Public Methods

- virtual void **setLambda** (real new\_lambda)  
*Change leak value on a leaky integrate function.*
- virtual void **setZeroPoint** (real new\_zeropoint)  
*Change zero point on a leaky integrate function.*
- virtual bool **processMessage** (ntime\_t t, const message\_base &m)  
*Processes message\_set\_lambda (p. 44) and message\_set\_zero\_point (p. 45) messages.*

### Protected Methods

- virtual void **zeropChange** ()

#### 10.8.1 Detailed Description

Abstract base class of a leaky integration function.

The function has a form of the following differential equation.  $\frac{dx}{dt} = f(t) - \lambda x$  The integration constant C is determined as  $f(\text{zerop})=0$  stands.

Definition at line 125 of file func.h.

#### 10.8.2 Member Function Documentation

##### 10.8.2.1 virtual void punnets\_common::func\_deriveq\_base::zeropChange () [inline, protected, virtual]

An entrance to recalculate coefficient at the zero-pointer change. In **func\_base** (p. 33) this function do nothing. This function will be defined in the derived classes with the requirement of the recalculation.

Definition at line 134 of file func.h.

Referenced by punnets\_common::func\_response::setZeroPoint(), and setZeroPoint().

```
134 { }
```

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

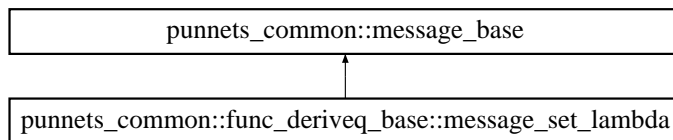
- **func.h**

## 10.9 punnets\_common::func\_deriveq\_base::message\_set\_lambda Class Reference

A message that changes lambda (leak value).

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_deriveq\_base::message\_set\_lambda::



### Public Methods

- virtual const char \* **getMessageId** () const

#### 10.9.1 Detailed Description

A message that changes lambda (leak value).

Definition at line 146 of file func.h.

#### 10.9.2 Member Function Documentation

##### 10.9.2.1 virtual const char\* punnets\_common::func\_deriveq\_base::message\_set\_lambda::getMessageId () [inline, virtual]

Returns the pointer of a string that represents the class. The pointer must be same for any instance of a given class. A function can check the type of the message by comparing the pointer.

Implements **punnets\_common::message\_base** (p. 68).

Definition at line 152 of file func.h.

```
152 { return messageId; }
```

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

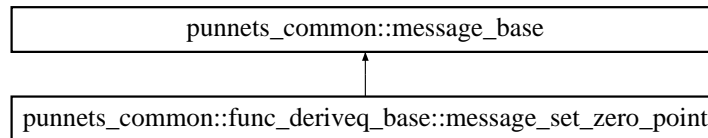
- **func.h**
- **func.cpp**

## 10.10 punnets\_common::func\_deriveq\_base::message\_set\_zero\_point Class Reference

A message that changes zero point.

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_deriveq\_base::message\_set\_zero\_point::



### Public Methods

- virtual const char \* `getMessageId ()` const

#### 10.10.1 Detailed Description

A message that changes zero point.

Definition at line 156 of file func.h.

#### 10.10.2 Member Function Documentation

##### 10.10.2.1 virtual const char\* punnets\_common::func\_deriveq\_base::message\_set\_zero\_point::getMessageId () [inline, virtual]

Returns the pointer of a string that represents the class. The pointer must be same for any instance of a given class. A function can check the type of the message by comparing the pointer.

Implements `punnets_common::message_base` (p. 68).

Definition at line 162 of file func.h.

```
162 { return messageId; }
```

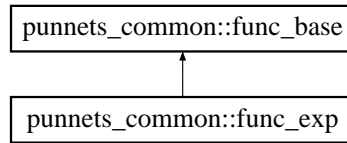
The documentation for this class was generated from the following files:

- `func.h`
- `func.cpp`

## 10.11 punnets\_common::func\_exp Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_exp:



### Public Methods

- **func\_exp** (real ir, real ipsi, real it0)
 

*Constructs a **func\_exp** (p. 46) with initial value r at time t0, and decay constant psi.*
- virtual **real getMaxGradient** (ntime\_t t) const
 

*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (ntime\_t t) const
 

*A pure virtual function that returns f(t).*
- virtual **real get1stDeriv** (ntime\_t t) const
 

*A pure virtual function that returns df(t)/dt.*
- virtual **real get2ndDeriv** (ntime\_t t) const
 

*A pure virtual function that returns d<sup>2</sup>f(t)/dt<sup>2</sup>.*
- virtual **ntime\_t getNextIncontinuity** (ntime\_t) const
- virtual **void getValueDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a zeroth-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **void get1stDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a first-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **void get2ndDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a second-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **func\_exp \* clone** ()
 

*Allocates and returns another instance of this function.*
- virtual **std::string getDescription** ()
 

*Obtains a human-readable description string of this function.*

### 10.11.1 Detailed Description

Exponential function

Exponentially decaying function  $f(t) = r \exp(-\psi; (t - t_0))$ .

Definition at line 655 of file func.h.

### 10.11.2 Member Function Documentation

#### 10.11.2.1 virtual ntime\_t punnets\_common::func\_exp::getNextIncontinuity (ntime\_t) const [inline, virtual]

A virtual function that returns the next incontinuity point after the time t. In **func\_base** (p. 33), the function always returns infinity.

Reimplemented from **punnets\_common::func\_base** (p. 34).

Definition at line 679 of file func.h.

References punnets\_common::ntime\_t.

```
679 { return mak::Infinity; };
```

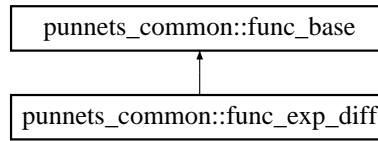
The documentation for this class was generated from the following file:

- **func.h**

## 10.12 punnets\_common::func\_exp\_diff Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_exp\_diff:



### Public Methods

- **func\_exp\_diff** (real ir1, real ipsi1, real ir2, real ipsi2, real it0)  
*constructs func\_exp\_diff* (p. 48).
- virtual **real getMaxGradient** (ntime\_t t) const  
*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (ntime\_t t) const  
*A pure virtual function that returns  $f(t)$ .*
- virtual **real get1stDeriv** (ntime\_t t) const  
*A pure virtual function that returns  $df(t)/dt$ .*
- virtual **real get2ndDeriv** (ntime\_t t) const  
*A pure virtual function that returns  $d^2f(t)/dt^2$ .*
- virtual **ntime\_t getNextIncontinuity** (ntime\_t) const
- virtual **void getValueDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const  
*Obtains a zeroth-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual **void get1stDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const  
*Obtains a first-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual **void get2ndDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const  
*Obtains a second-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual **func\_exp\_diff \* clone** ()  
*Allocates and returns another instance of this function.*
- virtual **std::string getDescription** ()  
*Obtains a human-readable description string of this function.*
- virtual **bool processMessage** (ntime\_t t, const message\_base &m)

### 10.12.1 Detailed Description

Difference of two exponential functions

External input of a form  $f(t) = r_1 \exp(-\psi_1(t - t_0)) - r_2 \exp(-\psi_2(t - t_0))$ . Here  $r_1, r_2 > 0$  and  $0 < \psi_1 < \psi_2$ .

Definition at line 801 of file func.h.

### 10.12.2 Member Function Documentation

#### 10.12.2.1 virtual ntime\_t punnets\_common::func\_exp\_diff::getNextIncontinuity (ntime\_t) const [inline, virtual]

A virtual function that returns the next incontinuity point after the time  $t$ . In **func\_base** (p. 33), the function always returns infinity.

Reimplemented from **punnets\_common::func\_base** (p. 34).

Definition at line 834 of file func.h.

References punnets\_common::ntime\_t.

```
834 { return mak::Infinity; }
```

#### 10.12.2.2 bool punnets\_common::func\_exp\_diff::processMessage (ntime\_t t, const message\_base & m) [virtual]

Process the specified message at the specified time. Return true if the message is processed. In **func\_base** (p. 33), this function always returns false (processes no message).

Reimplemented from **punnets\_common::func\_base** (p. 34).

Definition at line 529 of file func.cpp.

References punnets\_common::message\_base::getMessageId(), punnets\_common::ntime\_t, and punnets\_common::func\_base::valueChange().

```
530 {
531 #ifdef USE_DYNAMIC
532     if( dynamic_cast<const message_add_event_time *>(&m) != NULL )
533 #else
534     if( m.getMessageId() == message_add_event_time::messageId )
535 #endif
536     {
537         ir1 = r1orig + ir1 * exp( - psi1 * (t-it0) );
538         ir2 = r2orig + ir2 * exp( - psi2 * (t-it0) );
539         it0 = t;
540         valueChange();
541         return true;
542     }
543     return func_base::processMessage(t, m);
544 }
```

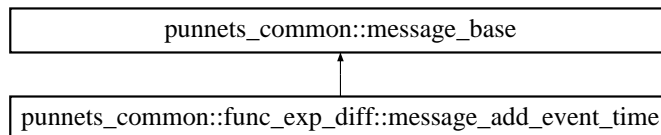
The documentation for this class was generated from the following files:

- **func.h**
- **func.cpp**

## 10.13 punnets\_common::func\_exp\_diff::message\_add\_event\_time Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_exp\_diff::message\_add\_event\_time::



### Public Methods

- virtual const char \* `getMessageId ()` const

#### 10.13.1 Detailed Description

Message to add an event. ????

Definition at line 846 of file func.h.

#### 10.13.2 Member Function Documentation

##### 10.13.2.1 virtual const char\* punnets\_common::func\_exp\_diff::message\_add\_event\_time::getMessageId () [inline, virtual]

Returns the pointer of a string that represents the class. The pointer must be same for any instance of a given class. A function can check the type of the message by comparing the pointer.

Implements `punnets_common::message_base` (p. 68).

Definition at line 851 of file func.h.

```
851 { return messageId; }
```

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

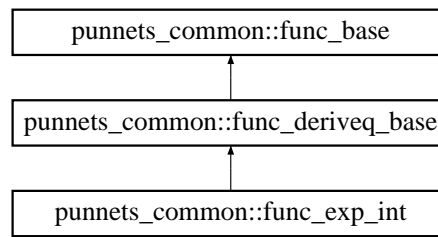
- `func.h`
- `func.cpp`



## 10.14 punnets\_common::func\_exp\_int Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_exp\_int::



### Public Methods

- **func\_exp\_int** (real ir, real ipsi, real it0)  
*Constructs a **func\_exp** (p. 46) with initial value r at time t0, and decay constant psi.*
- virtual **real getMaxGradient** (ntime\_t) const  
*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (ntime\_t t) const  
*A pure virtual function that returns f(t).*
- virtual **real get1stDeriv** (ntime\_t t) const  
*A pure virtual function that returns df(t)/dt.*
- virtual **real get2ndDeriv** (ntime\_t t) const  
*A pure virtual function that returns d<sup>2</sup>f(t)/dt<sup>2</sup>.*
- virtual **ntime\_t getNextIncontinuity** (ntime\_t) const
- virtual **void getValueDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const  
*Obtains a zeroth-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **void get1stDerivDomain** (ntime\_t, real &, real &, real &, real &) const  
*Obtains a first-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **void get2ndDerivDomain** (ntime\_t, real &, real &, real &, real &) const  
*Obtains a second-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **func\_exp\_int \* clone** ()  
*Allocates and returns another instance of this function.*
- virtual **std::string getDescription** ()  
*Obtains a human-readable description string of this function.*

### 10.14.1 Detailed Description

Integration of an exponential function

leaky integration of external, exponential input  $f(t) = r \exp(-\psi(t - t_0))$ .

**Todo:**

implement `get1stDerivDomain` etc. for this function

Definition at line 705 of file `func.h`.

### 10.14.2 Member Function Documentation

#### 10.14.2.1 `virtual ntime_t punnets_common::func_exp_int::getNextIncontinuity(ntime_t) const` [`inline`, `virtual`]

A virtual function that returns the next incontinuity point after the time  $t$ . In `func_base` (p. 33), the function always returns infinity.

Reimplemented from `punnets_common::func_base` (p. 34).

Definition at line 744 of file `func.h`.

References `punnets_common::ntime_t`.

```
744 { return mak::Infinity; };
```

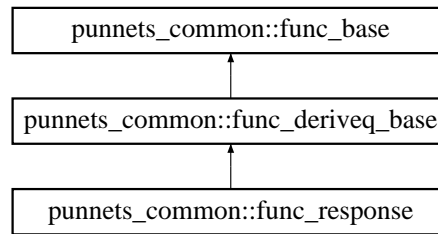
The documentation for this class was generated from the following file:

- `func.h`

## 10.15 punnets\_common::func\_response Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_response::



### Public Methods

- virtual **real getMaxGradient** (**ntime\_t** t) const  
*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (**ntime\_t** t) const  
*A pure virtual function that returns  $f(t)$ .*
- virtual **real get1stDeriv** (**ntime\_t** t) const  
*A pure virtual function that returns  $df(t)/dt$ .*
- virtual **real get2ndDeriv** (**ntime\_t** t) const  
*A pure virtual function that returns  $d^2f(t)/dt^2$ .*
- virtual **ntime\_t getNextIncontinuity** (**ntime\_t** t) const
- virtual void **getValueDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const  
*Obtains a zeroth-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get1stDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const  
*Obtains a first-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get2ndDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const  
*Obtains a second-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual std::string **getDescription** ()  
*Obtains a human-readable description string of this function.*
- virtual **func\_response \* clone** ()  
*Allocates and returns another instance of this function.*

- virtual void **setZeroPoint** (real new\_zeropoint)

*Change zero point on a leaky integrate function.*

### 10.15.1 Detailed Description

A response function.

**Todo:**

document this function

Definition at line 387 of file func.h.

### 10.15.2 Member Function Documentation

#### 10.15.2.1 virtual ntime\_t punnets\_common::func\_response::getNextIncontinuity (ntime\_t) const [inline, virtual]

A virtual function that returns the next incontinuity point after the time t. In **func\_base** (p. 33), the function always returns infinity.

Reimplemented from **punnets\_common::func\_base** (p. 34).

Definition at line 409 of file func.h.

References punnets\_common::ntime\_t.

```
409 { return mak::Infinity; };
```

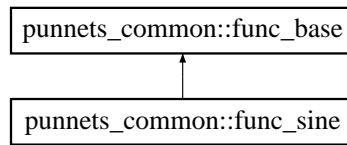
The documentation for this class was generated from the following files:

- **func.h**
- **func.cpp**

## 10.16 punnets\_common::func\_sine Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_sine::



### Public Methods

- **func\_sine** (real ir, real iomega, real itheta)
 

*Constructs a **func\_sine** (p.55) with amplitude r, angle velocity omega, and phase theta.*
- virtual **real getMaxGradient** (ntime\_t) const
 

*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (ntime\_t t) const
 

*A pure virtual function that returns f(t).*
- virtual **real get1stDeriv** (ntime\_t t) const
 

*A pure virtual function that returns df(t)/dt.*
- virtual **real get2ndDeriv** (ntime\_t t) const
 

*A pure virtual function that returns d<sup>2</sup>f(t)/dt<sup>2</sup>.*
- virtual **ntime\_t getNextIncontinuity** (ntime\_t) const
- virtual **void getValueDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a zeroth-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **void get1stDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a first-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **void get2ndDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a second-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **func\_sine \* clone** ()
 

*Allocates and returns another instance of this function.*
- virtual **std::string getDescription** ()
 

*Obtains a human-readable description string of this function.*

### 10.16.1 Detailed Description

Sinusoidal function

This function represents a sinusoidal function  $f(t) = r \sin(\omega t + \theta)$ .

Definition at line 431 of file func.h.

### 10.16.2 Member Function Documentation

#### 10.16.2.1 `virtual ntime_t punnets_common::func_sine::getNextIncontinuity` (`ntime_t`) const [inline, virtual]

A virtual function that returns the next incontinuity point after the time `t`. In `func_base` (p. 33), the function always returns infinity.

Reimplemented from `punnets_common::func_base` (p. 34).

Definition at line 460 of file func.h.

References `punnets_common::ntime_t`.

```
460 { return mak::Infinity; };
```

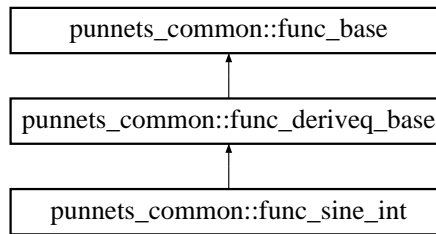
The documentation for this class was generated from the following files:

- `func.h`
- `func.cpp`

## 10.17 punnets\_common::func\_sine\_int Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_sine\_int::



### Public Methods

- **func\_sine\_int** (real ir, real iomega, real itheta)
 

*Constructs a **func\_sine\_int** (p.57) with amplitude r, angle velocity omega, and phase theta.*
- virtual **real getMaxGradient** (ntime\_t t) const
 

*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real getValue** (ntime\_t t) const
 

*A pure virtual function that returns f(t).*
- virtual **real get1stDeriv** (ntime\_t t) const
 

*A pure virtual function that returns df(t)/dt.*
- virtual **real get2ndDeriv** (ntime\_t t) const
 

*A pure virtual function that returns d<sup>2</sup>f(t)/dt<sup>2</sup>.*
- virtual void **getValueDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a zeroth-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual void **get1stDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a first-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual void **get2ndDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a second-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;downslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **func\_sine\_int \* clone** ()
 

*Allocates and returns another instance of this function.*
- virtual **std::string getDescription** ()

*Obtains a human-readable description string of this function.*

### 10.17.1 Detailed Description

Integrated function of **func\_sine** (p. 55)

This function represents an integration of a sinusoidal function.

Definition at line 548 of file func.h.

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

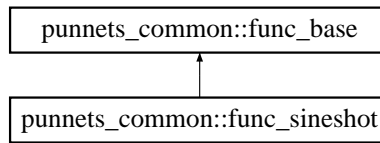
- **func.h**



## 10.18 punnets\_common::func\_sineshot Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_sineshot::



### Public Methods

- **func\_sineshot** (real ir, real iomega, real it0)
 

*Constructs a **func\_sineshot** (p. 59) with amplitude  $r$ , angle velocity  $\omega$ , and origin  $t_0$ .*
- virtual **real getMaxGradient** (ntime\_t t) const
 

*A pure virtual function that returns the max gradient of the function after the time  $t$ .*
- virtual bool **shouldDelete** (ntime\_t current)
 

*Return true if the function will return only zeros after the specified time.*
- virtual **real getValue** (ntime\_t t) const
 

*A pure virtual function that returns  $f(t)$ .*
- virtual **real get1stDeriv** (ntime\_t t) const
 

*A pure virtual function that returns  $df(t)/dt$ .*
- virtual **real get2ndDeriv** (ntime\_t t) const
 

*A pure virtual function that returns  $d^2f(t)/dt^2$ .*
- virtual **ntime\_t getNextIncontinuity** (ntime\_t from) const
- virtual void **getValueDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
- virtual void **get1stDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a first-order linear envelope of the function.  $\text{floor} < f(t+\mathcal{E}\alpha;) < \text{ceil}$ ,  $f(t)+\mathcal{E}\alpha;\text{downslope} < f(t+\mathcal{E}\alpha;) < f(t)+\mathcal{E}\alpha;\text{upslope}$ .*
- virtual void **get2ndDerivDomain** (ntime\_t t, real &upslope, real &ceil, real &downslope, real &floor) const
 

*Obtains a second-order linear envelope of the function.  $\text{floor} < f(t+\mathcal{E}\alpha;) < \text{ceil}$ ,  $f(t)+\mathcal{E}\alpha;\text{downslope} < f(t+\mathcal{E}\alpha;) < f(t)+\mathcal{E}\alpha;\text{upslope}$ .*
- virtual **func\_sineshot \* clone** ()
 

*Allocates and returns another instance of this function.*
- virtual std::string **getDescription** ()
 

*Obtains a human-readable description string of this function.*

- virtual bool `processMessage (ntime_t t, const message_base &m)`

### 10.18.1 Detailed Description

A single shot of sinusoidal function

This function represents a single shot (one cycle) of a sinusoidal function  $f(t) = r \sin^2(\omega(t - t_0))$  ( $t_0 \leq t \leq t_0 + \pi/\omega$ ).

Definition at line 477 of file `func.h`.

### 10.18.2 Member Function Documentation

#### 10.18.2.1 virtual ntime\_t punnets\_common::func\_sineshot::getNextIncontinuity (ntime\_t *from*) const [inline, virtual]

A virtual function that returns the next incontinuity point after the time  $t$ . In `func_base` (p. 33), the function always returns infinity.

Reimplemented from `punnets_common::func_base` (p. 34).

Definition at line 509 of file `func.h`.

References `punnets_common::ntime_t`.

```
509 { return from < t0 ? t0 : mak::Infinity; };
```

#### 10.18.2.2 void punnets\_common::func\_sineshot::getValueDomain (ntime\_t *t*, real & *upslope*, real & *ceil*, real & *downslope*, real & *floor*) const [virtual]

A virtual function that returns the next incontinuity point after the time  $t$ . Although this function has no incontinuity, it returns  $t_0$  because the linear envelope is divided at the point.

Implements `punnets_common::func_base` (p. 33).

Definition at line 175 of file `func.cpp`.

References `punnets_common::ntime_t`, and `punnets_common::real`.

```
176 {
177     upslope = ceil = downslope = floor = 0.0;
178     if( t < t0 || t >= t0 + duration )
179         return;
180     else
181     {
182         real phase = omega_2 * (t - t0);
183         real phi = phase + M_PI;
184         if( phi >= 2*M_PI ) phi -= 2*M_PI;
185         ( r > 0 ? ceil : floor ) = r;
186         if( phase < 0.5 * M_PI )
187         {
188             ( r > 0 ? upslope : downslope ) = omega_2 * r_div_2 * cos( alpha * cos( beta * phi ) + 0.5*M_PI - alpha);
189             ( r > 0 ? downslope : upslope ) = - omega_2 * r_div_2 * cos( alpha * cos( beta * (phi-M_PI) ) + 0.5*M_PI - alpha);
190         }
191         else if( phase < M_PI )
192         {
193             ( r > 0 ? upslope : downslope ) = r_div_2 * omega_2 * sin(phase );
```

```

194         ( r > 0 ? downslope : upslope ) = - omega_2 * r_div_2 * cos( alpha * cos( beta * (phi-M_PI) ) + 0.5*M_PI - a
195     }
196     else if( phase < 1.5*M_PI )
197     {
198         ( r > 0 ? downslope : upslope ) = - omega_2 * r_div_2 * cos( alpha * cos( beta * (phi+M_PI) ) + 0.5*M_PI - a
199     }
200     else
201     {
202         ( r > 0 ? downslope : upslope ) = r_div_2 * omega_2 * sin(phase);
203     }
204 }
205 }

```

### 10.18.2.3 virtual bool punnets\_common::func\_sineshot::processMessage (ntime\_t t, const message\_base & m) [inline, virtual]

Process the specified message at the specified time. Return true if the message is processed. In **func\_base** (p. 33), this function always returns false (processes no message).

Reimplemented from **punnets\_common::func\_base** (p. 34).

Definition at line 528 of file func.h.

References **punnets\_common::message\_base::getMessageId()**, and **punnets\_common::ntime\_t**.

```

528                                     {
529 #ifdef USE_DYNAMIC
530     if( dynamic_cast<const message_set_t0 *>(&m) != NULL )
531 #else
532     if( m.getMessageId() == message_set_t0::messageId )
533 #endif
534     {
535         t0 = t;
536         return true;
537     }
538     return func_base::processMessage(t, m);
539 }

```

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

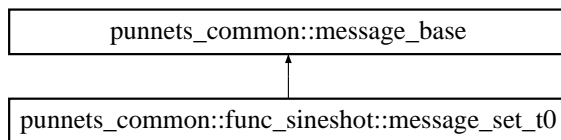
- **func.h**
- **func.cpp**

## 10.19 punnets\_common::func\_sineshot::message\_set\_t0 Class Reference

A message that changes t0.

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_sineshot::message\_set\_t0::



### Public Methods

- virtual const char \* **getMessageId** () const

#### 10.19.1 Detailed Description

A message that changes t0.

Definition at line 520 of file func.h.

#### 10.19.2 Member Function Documentation

##### 10.19.2.1 virtual const char\* punnets\_common::func\_sineshot::message\_set\_t0::getMessageId () [inline, virtual]

Returns the pointer of a string that represents the class. The pointer must be same for any instance of a given class. A function can check the type of the message by comparing the pointer.

Implements **punnets\_common::message\_base** (p. 68).

Definition at line 525 of file func.h.

```
525 { return messageId; }
```

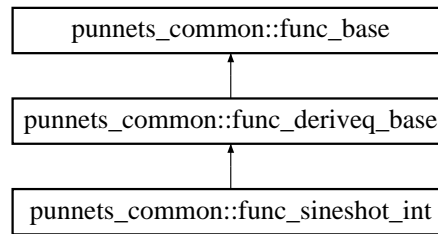
The documentation for this class was generated from the following files:

- **func.h**
- **func.cpp**

## 10.20 punnets\_common::func\_sineshot\_int Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::func\_sineshot\_int::



### Public Methods

- **func\_sineshot\_int** (**real** ir, **real** iomega)
  - Constructs a **func\_sineshot\_int** (p. 63) with amplitude r and angle velocity omega.*
- virtual **real** **getMaxGradient** (**ntime\_t** t) const
  - A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real** **getValue** (**ntime\_t** t) const
  - A pure virtual function that returns  $f(t)$ .*
- virtual **real** **get1stDeriv** (**ntime\_t** t) const
  - A pure virtual function that returns  $df(t)/dt$ .*
- virtual **real** **get2ndDeriv** (**ntime\_t** t) const
  - A pure virtual function that returns  $d^2f(t)/dt^2$ .*
- virtual void **getValueDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const
  - Obtains a zeroth-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get1stDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const
  - Obtains a first-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual void **get2ndDerivDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const
  - Obtains a second-order linear envelope of the function.  $floor < f(t+\mathcal{E}alpha;) < ceil$ ,  $f(t)+\mathcal{E}alpha;downslope < f(t+\mathcal{E}alpha;) < f(t)+\mathcal{E}alpha;upslope$ .*
- virtual **func\_sineshot\_int** \* **clone** ()
  - Allocates and returns another instance of this function.*
- virtual **std::string** **getDescription** ()

*Obtains a human-readable description string of this function.*

### 10.20.1 Detailed Description

Integrated function of **func\_sineshot** (p. 59)

This function represents an integration of **func\_sineshot** (p. 59). `t0` is always equal to zero.

Definition at line 612 of file `func.h`.

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

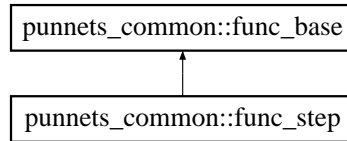
- **func.h**
- **func.cpp**

## 10.21 punnets\_common::func\_step Class Reference

</classdef>

#include <func.h>

Inheritance diagram for punnets\_common::func\_step::



### Public Methods

- virtual **real** **getMaxGradient** (**ntime\_t**) const  
*A pure virtual function that returns the max gradient of the function after the time t.*
- virtual **real** **getValue** (**ntime\_t** t) const  
*A pure virtual function that returns f(t).*
- virtual **real** **get1stDeriv** (**ntime\_t**) const  
*A pure virtual function that returns df(t)/dt.*
- virtual **real** **get2ndDeriv** (**ntime\_t**) const  
*A pure virtual function that returns d<sup>2</sup>f(t)/dt<sup>2</sup>.*
- virtual **ntime\_t** **getNextIncontinuity** (**ntime\_t** from) const
- virtual void **getValueDomain** (**ntime\_t** t, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const  
*Obtains a zeroth-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual void **get1stDerivDomain** (**ntime\_t**, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const  
*Obtains a first-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual void **get2ndDerivDomain** (**ntime\_t**, **real** &upslope, **real** &ceil, **real** &downslope, **real** &floor) const  
*Obtains a second-order linear envelope of the function. floor < f(t+ℰalpha;) < ceil, f(t)+ℰalpha;dwnslope < f(t+ℰalpha;) < f(t)+ℰalpha;upslope.*
- virtual **func\_step** \* **clone** ()  
*Allocates and returns another instance of this function.*
- virtual std::string **getDescription** ()  
*Obtains a human-readable description string of this function.*

### 10.21.1 Detailed Description

```
</classdef>
```

A step function.

This class represents a step function,  $f(t) = r s(t - t_0)$ .

Definition at line 277 of file func.h.

### 10.21.2 Member Function Documentation

#### 10.21.2.1 `virtual ntime_t punnets_common::func_step::getNextIncontinuity (ntime_t from) const [inline, virtual]`

A virtual function that returns the next incontinuity point after the time  $t$ . In `func_base` (p. 33), the function always returns infinity.

Reimplemented from `punnets_common::func_base` (p. 34).

Definition at line 294 of file func.h.

References `punnets_common::ntime_t`.

```
294 { return from < t0 ? t0 : mak::Infinity; };
```

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

- `func.h`



## 10.22 punnets\_common::greater\_tevent Struct Reference

```
#include <dsched.h>
```

### 10.22.1 Detailed Description

Function class to compare event times

This class is used to construct a local priority queue on STL. In a queue, events are sorted in an ascending order of event time. This class is privately used internally in the punnets library.

Definition at line 102 of file dsched.h.

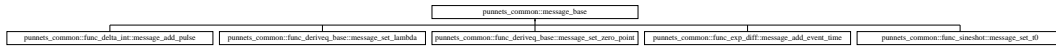
The documentation for this struct was generated from the following file:

- **dsched.h**

## 10.23 punnets\_common::message\_base Class Reference

```
#include <func.h>
```

Inheritance diagram for punnets\_common::message\_base::



### Public Methods

- virtual const char \* `getMessageId ()` const=0

#### 10.23.1 Detailed Description

Base class of a message to a function

A message is used to deliver a change to a function. Usually an event is delivered to a neuron, but a neuron has several functions. Moreover, we don't make neuron to handle correspondence of a specific event to a specific function. For this purpose a message is used, as every function knows which message has correspondence to itself.

Definition at line 49 of file func.h.

#### 10.23.2 Member Function Documentation

##### 10.23.2.1 virtual const char\* punnets\_common::message\_base::getMessageId () [pure virtual]

Returns the pointer of a string that represents the class. The pointer must be same for any instance of a given class. A function can check the type of the message by comparing the pointer.

Implemented in `punnets_common::func_deriveq_base::message_set_lambda` (p. 44), `punnets_common::func_deriveq_base::message_set_zero_point` (p. 45), `punnets_common::func_delta_int::message_add_pulse` (p. 42), `punnets_common::func_sineshot::message_set_t0` (p. 62), and `punnets_common::func_exp_diff::message_add_event_time` (p. 50).

Referenced by `punnets_common::func_sineshot::processMessage()`, `punnets_common::func_delta_int::processMessage()`, `punnets_common::func_deriveq_base::processMessage()`, and `punnets_common::func_exp_diff::processMessage()`.

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

- `func.h`

## 10.24 punnets\_common::taction Class Reference

```
#include <dsched.h>
```

Inheritance diagram for punnets\_common::taction::



### Public Methods

- virtual void **activate** (tscheduler &scheduler, **ntime\_t** current\_time)=0
- virtual tqueue \* **queue** () const=0  
*Obtain a local event queue of this action.*
- virtual const char \* **getClassName** () const=0  
*Get the class name of this action. Primarily for debugging.*

#### 10.24.1 Detailed Description

Action (some affection to an entity)

Class taction is a abstract base class of "changing something", such as pulse arrival etc. Event is an instance of an action, represented by a pair of time and action. When simulation time reaches the event time, the scheduler triggers the event; the corresponding action is "activated" to perform the change. E.g. when a pulse arrival is activated, the potential of the destination neuron is changed.

Punnets adopts a distributed queue model, in which every action has a corresponding "local queue".

#### Todo:

write more

Definition at line 52 of file dsched.h.

#### 10.24.2 Member Function Documentation

##### 10.24.2.1 virtual void punnets\_common::taction::activate (tscheduler & scheduler, ntime\_t current\_time) [pure virtual]

Activates the action at the specified time. When the corresponding new events are generated by the action, it is scheduled to the scheduler.

Implemented in **punnets\_common::tsentinel** (p.89), **punnets\_common::tlogger** (p.72), **punnets\_common::tsynapse\_base** (p.95), **punnets\_private::tsynapse< debug >** (p.90), **punnets\_private::tsynapse\_message< debug >** (p.98), **punnets\_private::tsynapse\_fatigue< debug >** (p.97), **punnets\_private::tneuron< debug >** (p.76), **punnets\_private::tneuron\_ext< debug >** (p.82), **punnets\_private::tsynapse\_addfunc< debug >** (p.92), and **punnets\_private::tsynapse\_messfunc< debug >** (p.101).

Referenced by punnets\_common::tevent::activate().

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

- `dsched.h`

## 10.25 punnets\_common::tevent Class Reference

```
#include <dsched.h>
```

### Public Methods

- **tevent** (**ntime\_t** itime, **taction** &iact)  
*construct an event with the specified time and action.*
- **ntime\_t** **getTime** () const  
*Obtain the time component of the event.*
- **taction** & **getAction** () const  
*Obtain the action component of the event.*
- void **activate** (tscheduler &scheduler) const  
*When the event time reaches, scheduler calls this member to trigger the event.*

### 10.25.1 Detailed Description

Event (scheduled action)

Class tevent represents a scheduled event, which is a pair of a scheduled time and an action. When simulation time reaches the event time, the scheduler triggers the event; the corresponding action is "activated" to perform the change.

Definition at line 74 of file dsched.h.

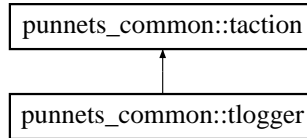
The documentation for this class was generated from the following file:

- **dsched.h**

## 10.26 punnets\_common::tlogger Class Reference

```
#include <dlogger.h>
```

Inheritance diagram for punnets\_common::tlogger::



### Public Types

- enum **logoption** { **shownone** = 0, **showthr** = 1, **showext** = 2, **showpart** = 4 }

### Public Methods

- **tlogger** (std::ostream &iout, **ntime\_t** istep, **ntime\_t** ifrom=0, **ntime\_t** iuntil=mak::Infinity)
 

*Constructs a logger with output stream iout, time step istep, and logging range between ifrom and iuntil.*
- virtual const char \* **getClassName** () const
 

*Get the class name of this action. Primarily for debugging.*
- virtual tqueue \* **queue** () const
 

*Obtain a local event queue of this action.*
- void **add** (tneuron\_base &p, **ntime\_t** delay, bool ishowthr, bool ishowext=false, **real** offset=0.0)
 

*Add a neuron to be logged. Two boolean specifies logging options of thresholds and externals.*
- void **add** (tneuron\_base &p, **ntime\_t** delay=0.0, **logoption** logopt=showthr, **real** offset=0.0)
 

*Add a neuron to be logged. logoption specifies logging options.*
- void **add** (tsynapse\_base &p, **ntime\_t** delay=0.0)
 

*Add a synapse to be logged.*
- virtual void **activate** (tscheduler &scheduler, **ntime\_t** current\_time)
 

*When activated by scheduler, the logger logs the current status to the log file.*
- void **schedule** (tscheduler &scheduler)
 

*Schedule the logger itself to the specified scheduler.*
- void **gnuplot\_def** (std::ostream &os, std::string file)

## 10.26.1 Detailed Description

Activation logging class.

This class periodically probes the state (potential value) of neurons, and produces a log file of the state changes. To use logging, you need to do the followings:

- Register the neurons to be logged by calling `add` method of the logger. If you want to plot several neurons separately, specify offsets to displace the graph vertically. Or you may specify the horizontal displacement by delay. This is useful for analyzing events with temporal delays.
- Schedule the logger to the scheduler by calling `schedule` method of the logger.
- Create gnuplot definition file by calling `gnuplot_def` method of the logger.

Definition at line 51 of file `dlogger.h`.

## 10.26.2 Member Enumeration Documentation

### 10.26.2.1 enum punnets\_common::tlogger::logoption

The option of the logging. You may log threshold, external inputs, and partitions. Multiple of them can be specified by logical or.

**Enumeration values:**

- shownone** Nothing.
- showthr** Threshold.
- showext** External inputs.
- showpart** Partitions.

Definition at line 58 of file `dlogger.h`.

Referenced by `add()`.

```

59     {
60         shownone = 0,
61         showthr = 1,
62         showext = 2,
63         showpart = 4
64     };

```

## 10.26.3 Member Function Documentation

### 10.26.3.1 void punnets\_common::tlogger::gnuplot\_def (std::ostream & *os*, std::string *file*)

Generate a GNUPLOT definition file to the specified stream. *file* is a file name of the log file.

Definition at line 90 of file `dlogger.cpp`.

References `showext`, `showpart`, and `showthr`.

```

91 {
92 // cerr << "min/max: " << delaymin << "/" << delaymax << endl;
93 // os << "set term X11" << endl;

```

```

94  os << "set y2tics" << endl;
95  os << "plot [" << from << ":" << until << "]"";
96
97  string delim = "";
98  int column = 1;
99  for( vector< neuentry >::iterator i = neus.begin(); i != neus.end(); i++ )
100 {
101     int xcol = (i->delay == 0.0 ? 1 : ++column);
102     os << delim << " \\" << endl << "\t'" << logfile << "' " <<
103         "using " << xcol << ":" << (++column) << " axes x1y1 " <<
104         "title \" " << i->neuron->getClassName() << " ' " << i->neuron->getName() << "'";
105     if( i->delay != 0.0 )
106         os << "(delay " << i->delay << ")";
107     if( i->offset != 0.0 )
108         os << "(offset " << i->offset << ")";
109     os << " signal\" with lines";
110     delim = ",";
111
112     if( i->logopt & showthr )
113     {
114         os << delim << " \\" << endl << "\t'" << logfile << "' " <<
115             "using " << xcol << ":" << (++column) << " axes x1y1 " <<
116             "title \" " << i->neuron->getClassName() << " ' " << i->neuron->getName() << "'";
117         if( i->delay != 0.0 )
118             os << "(delay " << i->delay << ")";
119         if( i->offset != 0.0 )
120             os << "(offset " << i->offset << ")";
121         os << " threshold\" with dots";
122         delim = ",";
123     }
124     if( i->logopt & showext )
125     {
126         os << delim << " \\" << endl << "\t'" << logfile << "' " <<
127             "using " << xcol << ":" << (++column) << " axes x1y1 " <<
128             "title \" " << i->neuron->getClassName() << " ' " << i->neuron->getName() << "'";
129         if( i->delay != 0.0 )
130             os << "(delay " << i->delay << ")";
131         if( i->offset != 0.0 )
132             os << "(offset " << i->offset << ")";
133         os << " external\" with dots";
134         delim = ",";
135     }
136     if( i->logopt & showpart )
137     {
138         os << delim << " \\" << endl << "\t'" << logfile << "' " <<
139             "using " << xcol << ":" << (++column) << " axes x1y1 " <<
140             "title \" " << i->neuron->getClassName() << " ' " << i->neuron->getName() << "'";
141         if( i->delay != 0.0 )
142             os << "(delay " << i->delay << ")";
143         if( i->offset != 0.0 )
144             os << "(offset " << i->offset << ")";
145         os << " partition 0\" with impulses";
146         delim = ",";
147
148         os << delim << " \\" << endl << "\t'" << logfile << "' " <<
149             "using " << xcol << ":" << (++column) << " axes x1y1 " <<
150             "title \" " << i->neuron->getClassName() << " ' " << i->neuron->getName() << "'";
151         if( i->delay != 0.0 )
152             os << "(delay " << i->delay << ")";
153         if( i->offset != 0.0 )
154             os << "(offset " << i->offset << ")";
155         os << " partition 1\" with impulses";
156
157         os << delim << " \\" << endl << "\t'" << logfile << "' " <<
158             "using " << xcol << ":" << (++column) << " axes x1y1 " <<
159             "title \" " << i->neuron->getClassName() << " ' " << i->neuron->getName() << "'";
160         if( i->delay != 0.0 )

```



```

161         os << "(delay " << i->delay << ")";
162     if( i->offset != 0.0 )
163         os << "(offset " << i->offset << ")";
164     os << " partition 2\" with impulses";
165
166     os << delim << " \\\" << endl << "\\t'" << logfile << "' " <<
167     "using " << xcol << ":" << (++column) << " axes x1y1 " <<
168     "title \"" << i->neuron->getClassname() << " '" << i->neuron->getName() << "'";
169     if( i->delay != 0.0 )
170         os << "(delay " << i->delay << ")";
171     if( i->offset != 0.0 )
172         os << "(offset " << i->offset << ")";
173     os << " partition delta_t\" with impulses";
174 }
175 }
176 for( vector<pair<tsynapse_base *, ntime_t> >::iterator i = syns.begin(); i != syns.end(); i++ )
177 {
178     int xcol = (i->second == 0.0 ? 1 : ++column);
179     os << delim << " \\\" << endl << "\\t'" << logfile << "' " <<
180     "using " << xcol << ":" << (++column) << " axes x1y2 " <<
181     "title \"" << i->first->getClassname() << " '" << i->first->getSrc().getName() << "' -> '" << i->first->getT
182     if( i->second != 0.0 )
183         os << "(delay " << i->second << ")";
184     os << "\" with lines";
185     delim = ", ";
186 }
187
188 os << endl;
189 os << "pause -1" << endl;
190 }

```

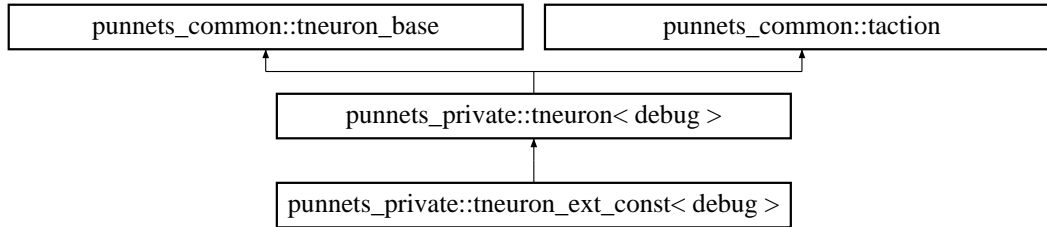
The documentation for this class was generated from the following files:

- [dlogger.h](#)
- [dlogger.cpp](#)

## 10.27 punnets\_private::tneuron< debug > Class Template Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tneuron< debug >::



### Public Methods

- **tneuron** (std::string iname="", ntime\_t isig\_hv\_period=def\_sig\_hv\_period, ntime\_t ithr\_hv\_period=def\_thr\_hv\_period, real imin\_threshold=def\_min\_threshold, real imax\_threshold=def\_max\_threshold)
- virtual real **getCurrentSigLevel** (ntime\_t current\_time)  
*Calculates the current signal level.*
- virtual real **getCurrentThrLevel** (ntime\_t current\_time)  
*Calculates the current threshold level.*
- virtual void **pulseArrive** (tscheduler &scheduler, ntime\_t current\_time, real pulse\_level)  
*Processes the pulse arrival.*
- virtual void **addSynapse** (tsynapse\_base \*s)  
*Add an synapse whose destination (post-synapse) is this neuron.*
- virtual void **eraseSynapse** (tsynapse\_base \*s)  
*Remove an synapse from this neuron.*
- virtual ntime\_t **getLastFire** () const  
*Get the time of the last firing.*
- virtual ntime\_t **getLastSimulate** () const  
*Get the time of the last simulation.*
- virtual void **activate** (tscheduler &scheduler, ntime\_t current\_time)  
*When activated as an event, it re-calculates its state and check firing.*
- virtual const char \* **getClassName** () const  
*Get the class name.*
- virtual tqueue \* **queue** () const  
*Get the pointer to the queue object.*

- const std::vector< tsynapse\_base \* > & **getSynapses** ()  
*Get the vector of synapses.*

## Protected Methods

- void **simulateElapse** (ntime\_t current\_time)  
*This function simulates time elapse up to the specified time.*
- virtual void **scheduleFire** (tscheduler &scheduler, ntime\_t current\_time, bool resched=true)  
*Schedule the next firing.*

## Protected Attributes

- ntime\_t **sig\_hv\_period**  
*Signal half-value period.*
- ntime\_t **thr\_hv\_period**  
*Threshold half-value period.*
- real **min\_threshold**  
*Minimum threshold level.*
- real **max\_threshold**  
*Maximum threshold level just after one firing.*
- real **coeff\_sigdecay**  
*Signal decaying coefficient is calculated from the signal half-value period.*
- real **coeff\_thrdecay**  
*Threshold decaying coefficient is calculated from the threshold half-value period.*

## Static Protected Attributes

- const real **sig\_converge\_level** = 0.0  
*The converge level (resting potential) of this neuron.*

### 10.27.1 Detailed Description

`template<bool debug> class punnets_private::tneuron< debug >`

The class defines a leaky integrate-and-fire neuron with threshold change, which receives only immediate pulses. A neuron itself behaves as an action of an event. When the next firing is predicted, the action is scheduled at the predicted firing time (called 'loopback'). When activated at a certain simulation time, the neuron re-calculates its state and process firing if necessary.

Definition at line 431 of file dneuron.h.

## 10.27.2 Constructor & Destructor Documentation

**10.27.2.1** `template<bool debug> punnets_private::tneuron< debug >::tneuron`  
 (`std::string iname = ""`, `ntime_t isig_hv_period = def_sig_hv_period`,  
`ntime_t ithr_hv_period = def_thr_hv_period`, `real imin_threshold =`  
`def_min_threshold`, `real imax_threshold = def_max_threshold`)

Construct a neuron with the specified name, signal half-value period, threshold half-value period, minimum threshold and maximum threshold.

Definition at line 122 of file `dneuron.cpp`.

References `punnets_common::ntime_t`, and `punnets_common::real`.

```

127         : tneuron_base(iname),
128           sig_level(0.0), last_simulate(0.0), last_fire(0.0),
129           sig_hv_period(isig_hv_period), thr_hv_period(ithr_hv_period),
130           min_threshold(imin_threshold), max_threshold(imax_threshold),
131           coeff_sigdecay(- M_LN2 / isig_hv_period),
132           coeff_thrdecay( - M_LN2 / ithr_hv_period )
133 { }
```

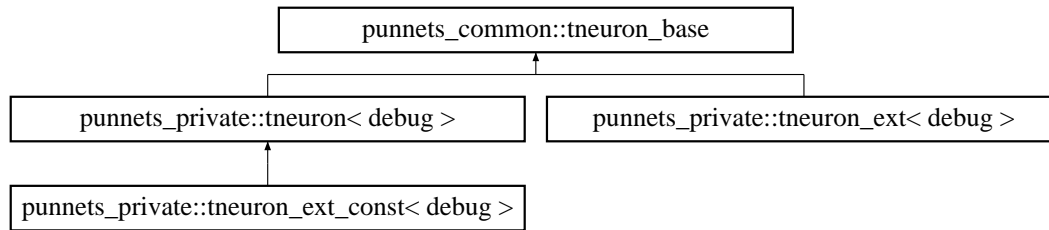
The documentation for this class was generated from the following files:

- `dneuron.h`
- `dneuron.cpp`

## 10.28 punnets\_common::tneuron\_base Class Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_common::tneuron\_base:



### Public Methods

- **tneuron\_base** (std::string iname="")  
*Construct a neuron with the name iname.*
- const std::string & **getName** () const  
*Returns the name of this neuron.*
- virtual void **pulseArrive** (tscheduler &scheduler, **ntime\_t** current\_time, **real** pulse\_level)=0
- virtual void **addSynapse** (**tsynapse\_base** \*)  
*Add a synapse input (this neuron becomes postsynaptic). A derived class should redefine this method.*
- virtual **real** **getCurrentSigLevel** (**ntime\_t**)  
*This function probes the current signal level of the neuron. A derived class should redefine this method.*
- virtual **real** **getCurrentThrLevel** (**ntime\_t**)  
*This function probes the current threshold level of the neuron. A derived class should redefine this method.*
- virtual **real** **getCurrentExtInput** (**ntime\_t**)  
*This function probes the current external input level of the neuron. A derived class should redefine this method.*
- virtual **ntime\_t** **getLastFire** () const  
*This function returns the last time that the neuron fired. Used in STDP and such.*
- virtual **ntime\_t** **getLastSimulate** () const  
*This function returns the last time that the neuron has been simulated.*
- virtual int **getLastSimulateType** () const  
*This function returns the type of the last simulation (0th/1st/2nd and so on).*
- virtual tqueue \* **queue** () const  
*Obtains the pointer to the queue object of this neuron.*

- virtual const char \* **getClassName** () const  
*Obtains the class name of this neuron.*

## Protected Attributes

- std::string **name**  
*The name of this neuron. Used in debugging and logging.*
- tqueue **\_queue**  
*The event queue of this neuron.*

### 10.28.1 Detailed Description

The base class of a neuron.

This abstract base class provides several interfaces for neuron access.

Definition at line 147 of file dneuron.h.

### 10.28.2 Member Function Documentation

#### 10.28.2.1 virtual void punnets\_common::tneuron\_base::pulseArrive (tscheduler & scheduler, ntime\_t current\_time, real pulse\_level) [pure virtual]

These methods handle pulse arrivals of the neuron. The pulse may be a real number (immediate value), message, or a function to be added. The latter two has a default handler that does nothing. A derived class should redefine these methods.

Implemented in **punnets\_private::tneuron< debug >** (p. 76), and **punnets\_private::tneuron\_ext< debug >** (p. 81).

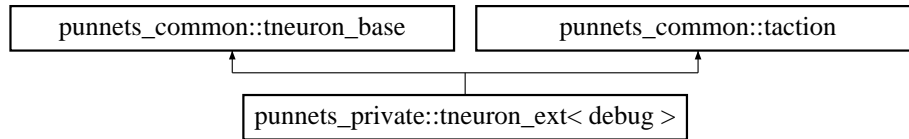
The documentation for this class was generated from the following file:

- **dneuron.h**

## 10.29 punnets\_private::tneuron\_ext< debug > Class Template Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tneuron\_ext< debug >::



### Public Methods

- **tneuron\_ext** (std::string iname="", real sig\_hv\_period=1.0)  
*Construct a **tneuron\_ext** (p. 81) class with the specified name and signal half-value period.*
- virtual real **getCurrentSigLevel** (ntime\_t current\_time)  
*Get the current signal level.*
- virtual real **getCurrentThrLevel** (ntime\_t)  
*Get the current threshold level, which is always 0.0.*
- virtual void **pulseArrive** (tscheduler &scheduler, ntime\_t current\_time, real pulse\_level)  
*Process the pulse arrival to this neuron.*
- virtual void **pulseArrive** (tscheduler &scheduler, ntime\_t current\_time, message\_base \*mess)  
*Process the message arrival to this neuron.*
- virtual void **pulseArrive** (tscheduler &scheduler, ntime\_t current\_time, func\_base \*func)  
*Process the function arrival to this neuron.*
- virtual void **addSynapse** (tsynapse\_base \*s)  
*Add an incoming synapse to this neuron.*
- virtual void **eraseSynapse** (tsynapse\_base \*s)  
*Erase an incoming synapse from this neuron.*
- virtual void **addExt** (func\_base \*s)  
*Add an external function to this neuron.*
- virtual ntime\_t **getLastFire** () const  
*This function returns the last time that the neuron fired. Used in STDP and such.*
- virtual ntime\_t **getLastSimulate** () const  
*This function returns the last time that the neuron has been simulated.*

- virtual int **getLastSimulateType** () const  
*This function returns the type of the last simulation (0th/1st/2nd and so on).*
- virtual void **activate** (tscheduler &scheduler, ntime\_t current\_time)  
*When activated as an event (loopback), it re-calculates its state and check firing.*
- virtual const char \* **getClassName** () const  
*Gets the class name.*
- const std::vector< tsynapse\_base \* > & **getSynapses** ()  
*Gets the collection of the synapses.*
- virtual tqueue \* **queue** () const  
*Get the queue object of this neuron.*
- void **setLoopBack** (tscheduler &scheduler, ntime\_t schedule\_time)

## Protected Methods

- void **fire** (tscheduler &scheduler, ntime\_t current\_time)  
*Process firing at the specified time.*
- real **calcSignal** (ntime\_t current\_time)  
*Calculate the power of signal at the specified time.*
- virtual void **scheduleFire** (tscheduler &scheduler, ntime\_t current\_time, bool resched=true)  
*Schedule the next firing.*
- bool **sendMessage** (ntime\_t t, message\_base \*mess)  
*Send the message to a function.*
- bool **broadcastMessage** (ntime\_t t, message\_base \*mess)  
*Send the message to all the functions.*

## Protected Attributes

- std::vector< tsynapse\_base \* > **synapses**  
*A collection of incoming synapses.*
- std::vector< func\_base \* > **exts**  
*A collection of functions.*
- func\_delta\_int \* **pulses**  
*A pulse function to support tsynapse.*
- ntime\_t **last\_simulate**  
*Last simulation time, only for simulate-logging.*



- `ntime_t last_fire`  
*Last firing time.*
- `ntime_t loopback`  
*Loopback time. It is used to check the loopback time changes,.*
- `real lambda`  
*The decaying coefficient for the inputs.*
- `int last_simulate_type`  
*The last simulation type, only for simulate-logging.*

## 10.29.1 Detailed Description

`template<bool debug> class punnets_private::tneuron_ext< debug >`

The extended neuron class. This class of neurons calculates the potential as a sum of functions. You can specify arbitrary functions, if you can provide the linear envelopes of the function and 1st/2nd derivatives.

Definition at line 604 of file dneuron.h.

## 10.29.2 Member Function Documentation

**10.29.2.1** `template<bool debug> void punnets_private::tneuron_ext< debug >::setLoopBack (tscheduler & scheduler, ntime_t schedule_time)`  
[inline]

Specify the loopback at the time. If the time is infinity, the loopback is cancelled. This method is public because the loopback is explicitly set.

Definition at line 718 of file dneuron.h.

References `punnets_common::ntime_t`.

Referenced by `punnets_private::tneuron_ext< debug >::scheduleFire()`.

```

719     {
720         loopback = schedule_time;
721         if( schedule_time < mak::Infinity )
722             scheduler.scheduleEvent( schedule_time, *this );
723     }
```

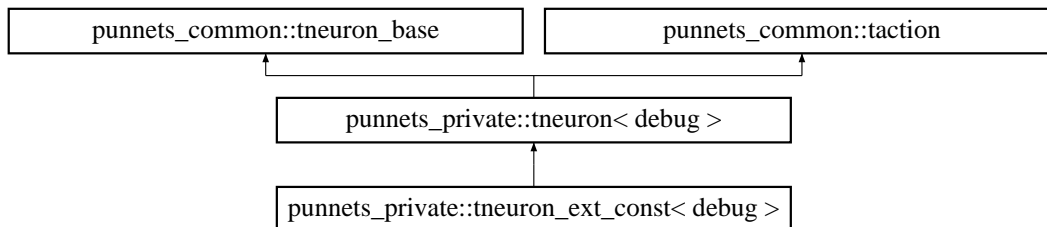
The documentation for this class was generated from the following files:

- `dneuron.h`
- `dneuron.cpp`

## 10.30 punnets\_private::tneuron\_ext\_const< debug > Class Template Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tneuron\_ext\_const< debug >::



### Public Methods

- **tneuron\_ext\_const** (std::string iname="", ntime\_t isig\_hv\_period=def\_sig\_hv\_period, ntime\_t ithr\_hv\_period=def\_thr\_hv\_period, real imin\_threshold=def\_min\_threshold, real imax\_threshold=def\_max\_threshold, real iext\_input=def\_ext\_input)
- virtual real **getCurrentSigLevel** (ntime\_t current\_time)
  - Calculates the current signal level.*
- real **getExtInput** ()
  - Get the weight of the external input.*
- virtual real **getCurrentExtInput** (ntime\_t)
  - Get the weight of the external input for logging.*
- void **setExtInput** (tscheduler &scheduler, ntime\_t current\_time, real val)
  - Change the weight of the external input at the specified time.*
- real **getConvergeLevel** ()
  - Get the converge level (resting potential of this neuron).*
- void **setConvergeLevel** (tscheduler &scheduler, ntime\_t current\_time, real val)
- virtual const char \* **getClassName** () const
  - Get the class name.*

### Protected Methods

- void **simulateElapse** (ntime\_t current\_time)
- virtual void **scheduleFire** (tscheduler &scheduler, ntime\_t current\_time, bool resched=true)
  - Schedule the next firing.*

## Protected Attributes

- real `sig_converge_level`

*The converge level (resting potential) of this neuron.*

### 10.30.1 Detailed Description

```
template<bool debug> class punnets_private::tneuron_ext_const< debug >
```

This class extends the `tneuron` class with a constant external input.

Definition at line 539 of file `dneuron.h`.

### 10.30.2 Constructor & Destructor Documentation

**10.30.2.1** `template<bool debug> punnets_private::tneuron_ext_const< debug >::tneuron_ext_const (std::string iname = "", ntime_t isig_hv_period = def_sig_hv_period, ntime_t ithr_hv_period = def_thr_hv_period, real imin_threshold = def_min_threshold, real imax_threshold = def_max_threshold, real iext_input = def_ext_input) [inline]`

Construct a neuron with the specified name, signal half-value period, threshold half-value period, minimum threshold, maximum threshold, and the weight of the external input.

Definition at line 562 of file `dneuron.h`.

References `punnets_common::ntime_t`, and `punnets_common::real`.

```
568         : tneuron<debug>( iname, isig_hv_period, ithr_hv_period, imin_threshold, imax_threshold ),
569           ext_input(iext_input), sig_converge_level(iext_input / coeff_sigdecay) { }
```

### 10.30.3 Member Function Documentation

**10.30.3.1** `template<bool debug> void punnets_private::tneuron_ext_const< debug >::setConvergeLevel (tscheduler & scheduler, ntime_t current_time, real val) [inline]`

Change the weight of the external input at the specified time, so that the converge level (resting potential of this neuron) becomes the specified value..

Definition at line 587 of file `dneuron.h`.

References `punnets_common::ntime_t`, and `punnets_common::real`.

```
588         { simulateElapse( current_time ); sig_converge_level = val; ext_input = sig_converge_level * coeff_sigdecay; sch
```

**10.30.3.2** `template<bool debug> void punnets_private::tneuron_ext_const< debug >::simulateElapse (ntime_t current_time) [inline, protected]`

This function simulates time elapse up to the specified time. Note that this function is non-virtual to speed-up the simulation.

Reimplemented from `punnets_private::tneuron< debug >` (p. 77).

Definition at line 548 of file `dneuron.h`.

References `punnets_common::ntime_t`.

```
549     {
550         sig_level = sig_converge_level +
551             ( ( sig_level - sig_converge_level )
552             * exp( (current_time - last_simulate) * coeff_sigdecay ) );
553         last_simulate = current_time;
554     }
```

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

- `dneuron.h`
- `dneuron.cpp`

## 10.31 tobserver Class Reference

### Public Methods

- **tobserver** (string iname, bool ishow=false)  
*The constructor.*
- virtual void **pulseArrive** (tscheduler &, ntime\_t current\_time, real pulse\_level)  
*When a pulse is arrived, it counts and reports the pulse arrival.*
- int **getNPulses** () const  
*Get the number of pulses observed.*
- void **clearNPulses** ()  
*Clear the pulse counter.*

#### 10.31.1 Detailed Description

An observer neuron class.

When this neuron received a pulse, it counts the number of pulses. If constructed with true *show* parameter, it also reports the arrival of a pulse to the console.

Definition at line 51 of file dttest.cpp.

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

- **dttest.cpp**

## 10.32 punnets\_common::tsched\_double Class Reference

```
#include <dsched.h>
```

### Public Methods

- **tsched\_double** ()  
*Constructs a scheduler with no events.*
- void **scheduleEvent** (const **tevent** &event)  
*Schedule an event.*
- void **scheduleEvent** (**ntime\_t** t, **taction** &act)  
*Schedule an action for the specified time.*
- bool **isScheduled** () const  
*Tests if the scheduler is not empty.*
- **ntime\_t** **run** (**ntime\_t** until=HUGE\_VAL)  
*Run the simulation (continue triggering events) until the specified time.*

### 10.32.1 Detailed Description

Event scheduler class.

The scheduler class activates events in an order of time.

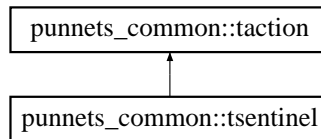
Definition at line 115 of file dsched.h.

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

- **dsched.h**
- **dsched.cpp**

## 10.33 punnets\_common::tsentinel Class Reference

Inheritance diagram for punnets\_common::tsentinel:



### Public Methods

- virtual void **activate** (tscheduler &, **ntime\_t**)
- tqueue \* **queue** () const  
*Obtain a local event queue of this action.*
- virtual const char \* **getClassname** () const  
*Get the class name of this action. Primarily for debugging.*

### 10.33.1 Detailed Description

Sentinel action.

This class is used to mark the end of the simulation range.

Definition at line 37 of file dsched.cpp.

### 10.33.2 Member Function Documentation

#### 10.33.2.1 virtual void punnets\_common::tsentinel::activate (tscheduler &, ntime\_t) [inline, virtual]

Activates the action at the specified time. When the corresponding new events are generated by the action, it is scheduled to the scheduler.

Implements **punnets\_common::taction** (p. 69).

Definition at line 46 of file dsched.cpp.

References punnets\_common::ntime\_t.

```

47     {
48         processed = true;
49     }
  
```

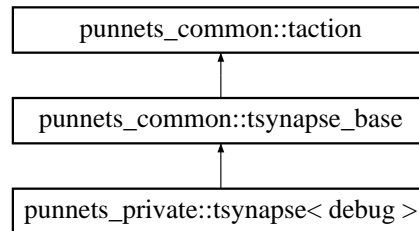
The documentation for this class was generated from the following file:

- **dsched.cpp**

## 10.34 punnets\_private::tsynapse< debug > Class Template Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tsynapse< debug >::



### Public Methods

- **tsynapse** (tneuron\_base &isrc, tneuron\_base &idest, ntime\_t idelay, real iweight)
  - Construct a synapse with the specified source, destination, delay and weight.*
- **tsynapse** (tneuron\_base &idest, ntime\_t idelay, real iweight)
- virtual void **setSrc** (tneuron\_base &isrc)
  - Specify the source neuron.*
- virtual real **getWeight** () const
  - Get the weight value of this synapse.*
- virtual tneuron\_base & **getSrc** () const
  - Get the source neuron (pre-synaptic).*
- virtual tneuron\_base & **getDest** () const
  - Get the destination neuron (post-synaptic).*
- void **addWeight** (real delta\_w)
  - Modify the weight value with the specified step size.*
- void **addDelay** (ntime\_t delta\_d)
  - Modify the delay with the specified step size.*
- virtual void **activate** (tscheduler &scheduler, ntime\_t current\_time)
  - The event handler delivers a pulse to the destination (post-synaptic) neuron.*
- virtual tqueue \* **queue** () const
  - Obtains the pointer to the queue object of this event.*



### 10.34.1 Detailed Description

`template<bool debug> class punnets_private::tsynapse< debug >`

The default synapse class. This class of synapses has an effect for the destination (post-synaptic) neuron to update its potential immediately with a constant amount of the weight.

Definition at line 265 of file dneuron.h.

### 10.34.2 Constructor & Destructor Documentation

**10.34.2.1** `template<bool debug> punnets_private::tsynapse< debug >::tsynapse  
(tneuron_base & idest, ntime_t idelay, real iweight) [inline]`

Construct a synapse with the specified destination, delay and weight. Source neuron will be specified after the construction via `setSrc()` (p. 90) method.

Definition at line 278 of file dneuron.h.

References `punnets_common::ntime_t`, and `punnets_common::real`.

```
279         : tsynapse_base(idelay), src(NULL), dest(&idest), weight(iweight) { }
```

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

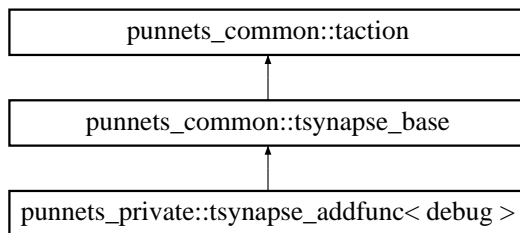
- `dneuron.h`

## 10.35 punnets\_private::tsynapse\_addfunc< debug > Class Template Reference

Synapse class that adds a new function to the destination (postsynaptic) **tneuron\_ext** (p. 81).

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tsynapse\_addfunc< debug >::



### Public Methods

- **tsynapse\_addfunc** (tneuron\_base &isrc, **tneuron\_ext**< debug > &idest, real idelay, func\_base \*ifunc, message\_base \*imess=NULL, real ilev=0.0)
- **tsynapse\_addfunc** (**tneuron\_ext**< debug > &idest, real idelay, func\_base \*ifunc, message\_base \*imess=NULL, real ilev=0.0)
- virtual void **setSrc** (tneuron\_base &isrc)

*Set the source neuron (presynaptic) of this synapse. A derived class should redefine this method.*

- virtual tneuron\_base & **getSrc** () const

*Get the source neuron (pre-synaptic). A derived class should redefine this method.*

- virtual **tneuron\_ext**< debug > & **getDest** () const

*Get the destination neuron (post-synaptic). A derived class should redefine this method.*

- virtual void **activate** (tscheduler &scheduler, ntime\_t current\_time)

*When activated, The synapse clones the function and added it to the destination neuron.*

- virtual tqueue \* **queue** () const

*Obtain a local event queue of this action.*

### 10.35.1 Detailed Description

```
template<bool debug> class punnets_private::tsynapse_addfunc< debug >
```

Synapse class that adds a new function to the destination (postsynaptic) **tneuron\_ext** (p. 81).

Definition at line 734 of file dneuron.h.

## 10.35.2 Constructor & Destructor Documentation

**10.35.2.1** `template<bool debug> punnets_private::tsynapse_addfunc< debug >::tsynapse_addfunc (tneuron_base & isrc, tneuron_ext< debug > & idest, real idelay, func_base * ifunc, message_base * imess = NULL, real ilev = 0.0) [inline]`

Constructs add-function synapse with the specified source, destination, delay, a pointer to a function, a pointer to a message that is sent before adding, and immediate pulse level.

Definition at line 746 of file dneuron.h.

References punnets\_common::real.

```
747      : tsynapse_base(idelay), src(&isrc), dest(&idest), func(ifunc), mess(imess), lev(ilev) { }
```

**10.35.2.2** `template<bool debug> punnets_private::tsynapse_addfunc< debug >::tsynapse_addfunc (tneuron_ext< debug > & idest, real idelay, func_base * ifunc, message_base * imess = NULL, real ilev = 0.0) [inline]`

Constructs add-function synapse with the specified destination, delay, a pointer to a function, a pointer to a message that is sent before adding, and immediate pulse level.

Definition at line 750 of file dneuron.h.

References punnets\_common::real.

```
751      : tsynapse_base(idelay), src(NULL), dest(&idest), func(ifunc), mess(imess), lev(ilev) { }
```

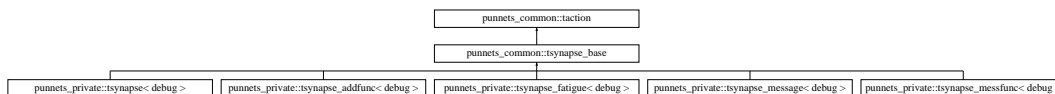
The documentation for this class was generated from the following file:

- dneuron.h

## 10.36 punnets\_common::tsynapse\_base Class Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_common::tsynapse\_base::



### Public Methods

- **tsynapse\_base** (**ntime\_t** idelay)  
*Construct a synapse with the specified delay.*
- **ntime\_t** **getDelay** () const  
*Get the delay length.*
- virtual **real** **getWeight** () const  
*Get the weight of the synapse. A derived class should redefine this method.*
- virtual void **setSrc** (**tneuron\_base** &)  
*Set the source neuron (presynaptic) of this synapse. A derived class should redefine this method.*
- virtual **tneuron\_base** & **getSrc** () const  
*Get the source neuron (pre-synaptic). A derived class should redefine this method.*
- virtual **tneuron\_base** & **getDest** () const  
*Get the destination neuron (post-synaptic). A derived class should redefine this method.*
- virtual void **activate** (**tscheduler** &scheduler, **ntime\_t** current\_time)=0
- virtual const char \* **getClassname** () const  
*Obtains the class name of this neuron.*

### Protected Attributes

- **ntime\_t** ndelay  
*synaptic delay.*

#### 10.36.1 Detailed Description

The base class of a synapse.

This abstract base class provides several interfaces for a synapse. A synapse itself behaves as an action of an event; i.e. when activated at a certain simulation time, a synapse updates the destination (post-synaptic) neuron.

Definition at line 217 of file dneuron.h.

## 10.36.2 Member Function Documentation

### 10.36.2.1 virtual void punnets\_common::tsynapse\_base::activate (tscheduler & scheduler, ntime\_t current\_time) [pure virtual]

The event handler. When the source (pre-synaptic) neuron fires, this function is called after the synaptic delay. A derived class should redefine this method to update the destination (post-synaptic) neuron.

Implements `punnets_common::taction` (p. 69).

Implemented in `punnets_private::tsynapse< debug >` (p. 90), `punnets_private::tsynapse_message< debug >` (p. 98), `punnets_private::tsynapse_fatigue< debug >` (p. 97), `punnets_private::tsynapse_addfunc< debug >` (p. 92), and `punnets_private::tsynapse_messfunc< debug >` (p. 101).

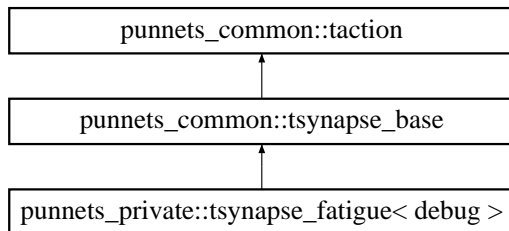
The documentation for this class was generated from the following file:

- `dneuron.h`

## 10.37 punnets\_private::tsynapse\_fatigue< debug > Class Template Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tsynapse\_fatigue< debug >::



### Public Methods

- **tsynapse\_fatigue** (tneuron\_base &isrc, tneuron\_base &idest, real idelay, real iweight)  
*Construct a synapse with the specified source, destination, delay and initial weight.*
- **tsynapse\_fatigue** (tneuron\_base &idest, real idelay, real iweight)
- virtual real **getWeight** () const  
*Get the weight value of this synapse.*
- virtual tneuron\_base & **getSrc** () const  
*Get the source neuron (pre-synaptic).*
- virtual tneuron\_base & **getDest** () const  
*Get the destination neuron (post-synaptic).*
- void **addWeight** (real delta\_w)  
*Modify the weight value with the specified step size.*
- void **addDelay** (ntime\_t delta\_d)  
*Modify the delay with the specified step size.*
- virtual void **activate** (tscheduler &scheduler, ntime\_t current\_time)
- virtual const char \* **getClassName** () const  
*Obtains the class name of this neuron.*

### Static Protected Attributes

- const ntime\_t **recover\_hv\_period** = 2000  
*The time of half-value period of the synapse weight decrease.*
- const real **fire\_ratio** = 0.002  
*The ratio of the decay of the weight caused by one pulse deliver.*

### 10.37.1 Detailed Description

template<bool debug> class punnets\_private::tsynapse\_fatigue< debug >

The "fatigue" synapse class. This class of synapses delivers a pulse to the destination (post-synaptic) neuron, whose strength decays on bursting (by lack of energy) and recovers gradually. This is a sample for constructing more complex synapse class.

Definition at line 363 of file dneuron.h.

### 10.37.2 Constructor & Destructor Documentation

**10.37.2.1** template<bool debug> punnets\_private::tsynapse\_fatigue< debug >::tsynapse\_fatigue (tneuron\_base & *idest*, real *idelay*, real *iweight*) [inline]

Construct a synapse with the specified destination, delay and initial weight. Source neuron will be specified after the construction via **setSrc()** (p. 94) method.

Definition at line 383 of file dneuron.h.

References punnets\_common::real.

```
384         : tsynapse_base(idelay), src(NULL), dest(&idest), weight(iweight / fire_ratio), last_fire( -Infinity ), last_weight(0)
```

### 10.37.3 Member Function Documentation

**10.37.3.1** template<bool debug> virtual void punnets\_private::tsynapse\_fatigue< debug >::activate (tscheduler & *scheduler*, ntime\_t *current\_time*) [inline, virtual]

The event handler delivers a pulse to the destination (post-synaptic) neuron. Then the last firing time is recorded to process the fatigueness.

Implements **punnets\_common::tsynapse\_base** (p. 95).

Definition at line 401 of file dneuron.h.

References punnets\_common::ntime\_t, and punnets\_common::real.

```
402     {
403         real this_weight = weight + (last_weight - weight) * exp( (current_time - last_fire) * (-M_LN2 / recover_hv_peri) );
404         if( getDeb() )
405             std::cout << std::setw(9) << std::setiosflags(std::ios::fixed) << std::setprecision(debug_precision) << current_time << "\n";
406         " Pulse Arrived from " << src->getName() << "(" << this_weight * fire_ratio << "/" << weight << ")" << std::endl;
407         dest->pulseArrive( scheduler, current_time, this_weight * fire_ratio );
408         last_fire = current_time;
409         last_weight = this_weight * (1.0 - fire_ratio);
410     }
```

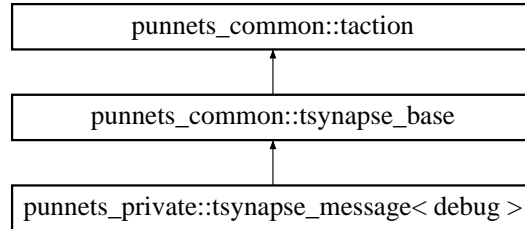
The documentation for this class was generated from the following file:

- dneuron.h

## 10.38 punnets\_private::tsynapse\_message< debug > Class Template Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tsynapse\_message< debug >::



### Public Methods

- **tsynapse\_message** (tneuron\_base &isrc, tneuron\_base &idest, real idelay, message\_base \*imess)
- **tsynapse\_message** (tneuron\_base &idest, real idelay, message\_base \*imess)
- virtual **~tsynapse\_message** ()  
*Destructor deletes the message.*
- virtual void **setSrc** (tneuron\_base &isrc)  
*Specify the source neuron.*
- virtual tneuron\_base & **getSrc** () const  
*Get the source neuron (pre-synaptic).*
- virtual tneuron\_base & **getDest** () const  
*Get the destination neuron (post-synaptic).*
- virtual void **activate** (tscheduler &scheduler, ntime\_t current\_time)  
*The event handler delivers a message to the destination (post-synaptic) neuron.*

### 10.38.1 Detailed Description

```
template<bool debug> class punnets_private::tsynapse_message< debug >
```

The message synapse class. This class of synapses has an effect for the destination (post-synaptic) neuron to deliver a message. The effect of the message can be arbitrarily specified.

Definition at line 317 of file dneuron.h.



## 10.38.2 Constructor & Destructor Documentation

### 10.38.2.1 `template<bool debug> punnets_private::tsynapse_message< debug >::tsynapse_message (tneuron_base & isrc, tneuron_base & idest, real idelay, message_base * imess) [inline]`

Construct a synapse with the specified source, destination, delay and a pointer to a message. The pointer to the message will be deleted at the destruction of this synapse.

Definition at line 327 of file dneuron.h.

References `punnets_common::real`.

```
328         : tsynapse_base(idelay), src(&isrc), dest(&idest), mess(imess) { }
```

### 10.38.2.2 `template<bool debug> punnets_private::tsynapse_message< debug >::tsynapse_message (tneuron_base & idest, real idelay, message_base * imess) [inline]`

Construct a synapse with the specified destination, delay and a pointer to a message. Source neuron will be specified after the construction via `setSrc()` (p.98) method. The pointer to the message will be deleted at the destruction of this synapse.

Definition at line 332 of file dneuron.h.

References `punnets_common::real`.

```
333         : tsynapse_base(idelay), src(NULL), dest(&idest), mess(imess) { }
```

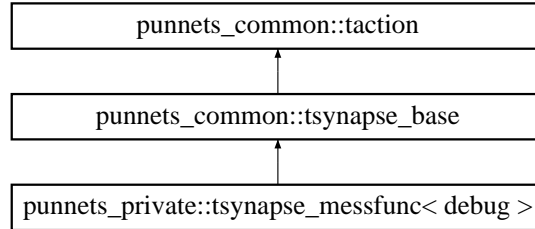
The documentation for this class was generated from the following file:

- `dneuron.h`

## 10.39 punnets\_private::tsynapse\_messfunc< debug > Class Template Reference

```
#include <dneuron.h>
```

Inheritance diagram for punnets\_private::tsynapse\_messfunc< debug >::



### Public Methods

- **tsynapse\_messfunc** (tneuron\_base &isrc, **tneuron\_ext**< debug > &idest, real idelay, func\_base \*ifunc, message\_base \*imess, real ilev=0.0)
- **tsynapse\_messfunc** (**tneuron\_ext**< debug > &idest, real idelay, func\_base \*ifunc, message\_base \*imess, real ilev=0.0)
- virtual void **setSrc** (tneuron\_base &isrc)

*Set the source neuron (presynaptic) of this synapse. A derived class should redefine this method.*

- virtual tneuron\_base & **getSrc** () const

*Get the source neuron (pre-synaptic). A derived class should redefine this method.*

- virtual **tneuron\_ext**< debug > & **getDest** () const

*Get the destination neuron (post-synaptic). A derived class should redefine this method.*

- virtual void **activate** (tscheduler &scheduler, ntime\_t current\_time)
- virtual tqueue \* **queue** () const

*Obtain a local event queue of this action.*

### 10.39.1 Detailed Description

```
template<bool debug> class punnets_private::tsynapse_messfunc< debug >
```

Synapse class that sends a message to a function in a **tneuron\_ext** (p.81) neuron. It is useful for applying an effect of `func_expdiff`, on which several firing can be calculated by one function and `message_add_event_time` message.

Definition at line 779 of file `dneuron.h`.

## 10.39.2 Constructor & Destructor Documentation

**10.39.2.1** `template<bool debug> punnets_private::tsynapse_messfunc< debug >::tsynapse_messfunc (tneuron_base & isrc, tneuron_ext< debug > & idest, real idelay, func_base * ifunc, message_base * imess, real ilev = 0.0) [inline]`

Constructs add-function synapse with the specified source, destination, delay, a pointer to a function, a pointer to a message that is sent for the function at every activation, and immediate pulse level. In the constructor the function is added to the destination neuron.

Definition at line 792 of file dneuron.h.

References `punnets_private::tneuron_ext< debug >::addExt()`, and `punnets_common::real`.

```
793      : tsynapse_base(idelay), src(&isrc), dest(&idest), func(ifunc), mess(imess), lev(ilev) { dest->addExt(func); }
```

**10.39.2.2** `template<bool debug> punnets_private::tsynapse_messfunc< debug >::tsynapse_messfunc (tneuron_ext< debug > & idest, real idelay, func_base * ifunc, message_base * imess, real ilev = 0.0) [inline]`

Constructs add-function synapse with the specified destination, delay, a pointer to a function, a pointer to a message that is sent for the function at every activation, and immediate pulse level. In the constructor the function is added to the destination neuron.

Definition at line 797 of file dneuron.h.

References `punnets_private::tneuron_ext< debug >::addExt()`, and `punnets_common::real`.

```
798      : tsynapse_base(idelay), src(NULL), dest(&idest), func(ifunc), mess(imess), lev(ilev) { dest->addExt(func); }
```

## 10.39.3 Member Function Documentation

**10.39.3.1** `template<bool debug> virtual void punnets_private::tsynapse_messfunc< debug >::activate (tscheduler & scheduler, ntime_t current_time) [inline, virtual]`

When activated, the synapse sends the message to the function and sends the pulse to the destination neuron (to notice the change of the function).

Implements `punnets_common::tsynapse_base` (p. 95).

Definition at line 809 of file dneuron.h.

References `punnets_common::tneuron_base::getName()`, `punnets_common::ntime_t`, and `punnets_private::tneuron_ext< debug >::pulseArrive()`.

```
810    {
811        if( getDeb() )
812            std::cout << std::setw(9) << std::setiosflags(std::ios::fixed) << std::setprecision(debug_precision) << cur
813            func->processMessage( current_time, *mess );
814        // cout << f->getDescription() << endl;
815        dest->pulseArrive( scheduler, current_time, lev );
816    }
```

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

- `dneuron.h`



# Chapter 11

## Punnets File Documentation

### 11.1 dlanguage.cpp File Reference

Language simulation based on the punnets library.

```
#include <stdlib.h>
#include <sstream>
#include <fstream>
#include <iomanip>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <math.h>
#include <string>
#include <mak/hash_string.h>
#include "punnets.h"
```

#### Namespaces

- namespace `__gnu_cxx`
- namespace `mak`
- namespace `std`

#### Compounds

- struct `neuinfo`
- struct `word_t`

#### 11.1.1 Detailed Description

Language simulation based on the punnets library.

See the paper “A Pulsed Neural Network for Language Understanding: Discrete-Event Simulation of a Short-Term Memory Mechanism and Sentence Understanding” for details. In short, we assign one representer neuron (*exts*) to represent each word. Once fired, a representer neuron periodically fires to keep short-term memory of the word. Representer neurons are interconnected via two networks, named autoassociative network (*sgates*) and heteroassociative network (*dgates*). When the network receives several words in a sequence, the two networks compute the bindings of the words, and as a result, the meaning of the input sentence is represented by the activation pattern of the representer neurons (synchronized neurons have been bound). The connection weights of the two networks are obtained from the external files.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:****Id:**

dlanguage.cpp,v 1.7 2003/05/08 08:23:56 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file **dlanguage.cpp**.

## 11.1.2 Variable Documentation

### 11.1.2.1 `const string posstrs[]`

**Initial value:**

```
{ "pronoun 1st", "pronoun 3rd", "noun", "proper noun",
  "vi", "vt", "vt_subj", "det" }
```

Definition at line 104 of file `dlanguage.cpp`.

## 11.2 dlogger.cpp File Reference

Logger.

```
#include "dlogger.h"  
#include <iomanip>
```

### Namespaces

- namespace `punnets_common`

### 11.2.1 Detailed Description

Logger.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

dlogger.cpp,v 1.1 2003/05/01 10:57:43 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file `dlogger.cpp`.

## 11.3 dlogger.h File Reference

Activation logging class.

```
#include "dneuron.h"  
#include <mak/infinity.h>
```

### Namespaces

- namespace `punnets_common`

### 11.3.1 Detailed Description

Activation logging class.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

dlogger.h,v 1.2 2003/05/08 07:24:56 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file **dlogger.h**.



## 11.4 dneuron.cpp File Reference

Neurons.

```
#include "dneuron.h"
#include <math.h>
#include <mak/profile.h>
```

### Namespaces

- namespace **punnets\_common**
- namespace **punnets\_private**

### 11.4.1 Detailed Description

Neurons.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

dneuron.cpp,v 1.7 2003/05/07 09:32:23 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file **dneuron.cpp**.

## 11.5 dneuron.h File Reference

Neuron/Synapse class in discrete-event NN simulation.

```
#include <vector>
#include <map>
#include <string>
#include <algorithm>
#include <math.h>
#include <queue>
#include <iostream>
#include <iomanip>
#include "dsched.h"
#include "func.h"
```

### Namespaces

- namespace **punnets**
- namespace **punnets\_common**
- namespace **punnets\_nodebug**
- namespace **punnets\_private**

### 11.5.1 Detailed Description

Neuron/Synapse class in discrete-event NN simulation.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

dneuron.h,v 1.6 2003/05/08 07:24:56 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file **dneuron.h**.

## 11.6 dsched.cpp File Reference

Event, Action, Scheduler.

```
#include "dsched.h"
```

### Namespaces

- namespace **punnets\_common**

### 11.6.1 Detailed Description

Event, Action, Scheduler.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

dsched.cpp,v 1.1 2003/05/01 10:57:43 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file **dsched.cpp**.

## 11.7 dsched.h File Reference

Distributed scheduler.

```
#include <math.h>
#include <functional>
#include <map>
#include <set>
#include <vector>
#include <queue>
#include <string>
#include "punnets_base.h"
```

### Namespaces

- namespace `punnets_common`

#### 11.7.1 Detailed Description

Distributed scheduler.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

dsched.h,v 1.2 2003/05/08 07:24:56 t Exp

Copyright (C) 2003 Makino, Takaki. All rights reserved.

Definition in file `dsched.h`.

## 11.8 dtest.cpp File Reference

Test program of the punnets library.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include "dneuron.h"
#include "dsched.h"
#include "dlogger.h"
#include <mak/cmdopt.h>
```

### Compounds

- class `tobserver`

#### 11.8.1 Detailed Description

Test program of the punnets library.

This program tests the performance of delayed firing simulations. Every neuron has an sinusoidal external input and interconnecting synapses, which causes alpha-function-style (difference of two exponential functions) response. Two hundred pulses are injected to produce initial activity. You can change parameters by command-line options; try `dtest -h` to show the list of options.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

dtest.cpp,v 1.7 2003/05/08 08:23:56 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file `dtest.cpp`.

## 11.9 func.cpp File Reference

Functions.

```
#include "func.h"  
#include <iostream>
```

### Namespaces

- namespace `punnets_common`

### 11.9.1 Detailed Description

Functions.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

func.cpp,v 1.3 2003/05/02 09:14:21 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file `func.cpp`.

## 11.10 func.h File Reference

Function representation.

```
#include <math.h>
#include <string>
#include <iostream>
#include <sstream>
#include <mak/infinity.h>
#include "punnets_base.h"
```

### Namespaces

- namespace `punnets_common`

#### 11.10.1 Detailed Description

Function representation.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

func.h,v 1.2 2003/05/02 09:13:05 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file `func.h`.

## 11.11 punnets.h File Reference

Punnets header file.

```
#include "config_punnets.h"
#include "func.h"
#include "dsched.h"
#include "dlogger.h"
#include "dneuron.h"
```

### 11.11.1 Detailed Description

Punnets header file.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

punnets.h,v 1.4 2003/05/08 08:23:56 t Exp

Copyright (C) 2003 Makino, Takaki.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Definition in file **punnets.h**.



## 11.12 punnets\_base.h File Reference

Base class of Punnets.

```
#include "config_punnets.h"
```

### Namespaces

- namespace **punnets**
- namespace **punnets\_common**
- namespace **punnets\_nodebug**
- namespace **punnets\_private**

### 11.12.1 Detailed Description

Base class of Punnets.

**Author:**

Makino, Takaki <t-makino-punnets01@snowelm.com>

**Date:**

2003-05-01

**Version:**

**Id:**

punnets\_base.h,v 1.2 2003/05/08 07:24:56 t Exp

Copyright (C) 2003 Makino, Takaki. All rights reserved.

Definition in file **punnets\_base.h**.



## Chapter 12

# Punnets Page Documentation

### 12.1 Todo List

Class `punnets_common::func_exp_int` implement `get1stDerivDomain` etc. for this function

Class `punnets_common::func_response` document this function

Class `punnets_common::taction` write more

# Index

- `_queue`
  - `punnets_common::neuron_base`, 80
- `~tsynapse_message`
  - `punnets_private::tsynapse_message`, 98
- `activate`
  - `punnets_common::taction`, 69
  - `punnets_common::tevent`, 71
  - `punnets_common::tlogger`, 72
  - `punnets_common::tsentinel`, 89
  - `punnets_common::tsynapse_base`, 95
  - `punnets_private::neuron`, 76
  - `punnets_private::neuron_ext`, 82
  - `punnets_private::tsynapse`, 90
  - `punnets_private::tsynapse_addfunc`, 92
  - `punnets_private::tsynapse_fatigue`, 97
  - `punnets_private::tsynapse_message`, 98
  - `punnets_private::tsynapse_messfunc`, 101
- `add`
  - `punnets_common::tlogger`, 72
- `addDelay`
  - `punnets_private::tsynapse`, 90
  - `punnets_private::tsynapse_fatigue`, 96
- `addExt`
  - `punnets_private::neuron_ext`, 81
- `addSynapse`
  - `punnets_common::neuron_base`, 79
  - `punnets_private::neuron`, 76
  - `punnets_private::neuron_ext`, 81
- `addWeight`
  - `punnets_private::tsynapse`, 90
  - `punnets_private::tsynapse_fatigue`, 96
- `broadcastMessage`
  - `punnets_private::neuron_ext`, 82
- `calcSignal`
  - `punnets_private::neuron_ext`, 82
- `clearNPulses`
  - `tobserver`, 87
- `clone`
  - `punnets_common::func_base`, 33
  - `punnets_common::func_const`, 36
  - `punnets_common::func_const_int`, 38
  - `punnets_common::func_delta_int`, 41
  - `punnets_common::func_exp`, 46
  - `punnets_common::func_exp_diff`, 48
  - `punnets_common::func_exp_int`, 51
  - `punnets_common::func_response`, 53
  - `punnets_common::func_sine`, 55
  - `punnets_common::func_sine_int`, 57
  - `punnets_common::func_sineshot`, 59
  - `punnets_common::func_sineshot_int`, 63
  - `punnets_common::func_step`, 65
- `coeff_sigdecay`
  - `punnets_private::neuron`, 77
- `coeff_thrdecay`
  - `punnets_private::neuron`, 77
- `debug`
  - `punnets_common::debugflag`< `true` >, 32
- `debugflag`
  - `punnets_common::debugflag`< `true` >, 32
- `dlanguage.cpp`, 103
  - `posstrs`, 104
- `dlogger.cpp`, 105
- `dlogger.h`, 106
- `dneuron.cpp`, 107
- `dneuron.h`, 108
- `dsched.cpp`, 109
- `dsched.h`, 110
- `dtest.cpp`, 111
- `epsilon`
  - `punnets_common`, 25
- `eraseSynapse`
  - `punnets_private::neuron`, 76
  - `punnets_private::neuron_ext`, 81
- `exts`
  - `punnets_private::neuron_ext`, 82
- `fire`
  - `punnets_private::neuron_ext`, 82
- `fire_ratio`
  - `punnets_private::tsynapse_fatigue`, 96
- `func.cpp`, 112
- `func.h`, 113
- `func_const`

- punnets\_common::func\_const, 36
- func\_const\_int
  - punnets\_common::func\_const\_int, 38
- func\_delta\_int
  - punnets\_common::func\_delta\_int, 40
- func\_exp
  - punnets\_common::func\_exp, 46
- func\_exp\_diff
  - punnets\_common::func\_exp\_diff, 48
- func\_exp\_int
  - punnets\_common::func\_exp\_int, 51
- func\_sine
  - punnets\_common::func\_sine, 55
- func\_sine\_int
  - punnets\_common::func\_sine\_int, 57
- func\_sineshot
  - punnets\_common::func\_sineshot, 59
- func\_sineshot\_int
  - punnets\_common::func\_sineshot\_int, 63
- get1stDeriv
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 40
  - punnets\_common::func\_exp, 46
  - punnets\_common::func\_exp\_diff, 48
  - punnets\_common::func\_exp\_int, 51
  - punnets\_common::func\_response, 53
  - punnets\_common::func\_sine, 55
  - punnets\_common::func\_sine\_int, 57
  - punnets\_common::func\_sineshot, 59
  - punnets\_common::func\_sineshot\_int, 63
  - punnets\_common::func\_step, 65
- get1stDerivDomain
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 40
  - punnets\_common::func\_exp, 46
  - punnets\_common::func\_exp\_diff, 48
  - punnets\_common::func\_exp\_int, 51
  - punnets\_common::func\_response, 53
  - punnets\_common::func\_sine, 55
  - punnets\_common::func\_sine\_int, 57
  - punnets\_common::func\_sineshot, 59
  - punnets\_common::func\_sineshot\_int, 63
  - punnets\_common::func\_step, 65
- get2ndDeriv
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 40
  - punnets\_common::func\_exp, 46
- punnets\_common::func\_exp\_diff, 48
- punnets\_common::func\_exp\_int, 51
- punnets\_common::func\_response, 53
- punnets\_common::func\_sine, 55
- punnets\_common::func\_sine\_int, 57
- punnets\_common::func\_sineshot, 59
- punnets\_common::func\_sineshot\_int, 63
- punnets\_common::func\_step, 65
- get2ndDerivDomain
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 40
  - punnets\_common::func\_exp, 46
  - punnets\_common::func\_exp\_diff, 48
  - punnets\_common::func\_exp\_int, 51
  - punnets\_common::func\_response, 53
  - punnets\_common::func\_sine, 55
  - punnets\_common::func\_sine\_int, 57
  - punnets\_common::func\_sineshot, 59
  - punnets\_common::func\_sineshot\_int, 63
  - punnets\_common::func\_step, 65
- getAction
  - punnets\_common::tevent, 71
- getClassname
  - punnets\_common::taction, 69
  - punnets\_common::tlogger, 72
  - punnets\_common::tneuron\_base, 80
  - punnets\_common::tsentinel, 89
  - punnets\_common::tsynapse\_base, 94
  - punnets\_private::tneuron, 76
  - punnets\_private::tneuron\_ext, 82
  - punnets\_private::tneuron\_ext\_const, 84
  - punnets\_private::tsynapse\_fatigue, 96
- getConvergeLevel
  - punnets\_private::tneuron\_ext\_const, 84
- getCurrentExtInput
  - punnets\_common::tneuron\_base, 79
  - punnets\_private::tneuron\_ext\_const, 84
- getCurrentSigLevel
  - punnets\_common::tneuron\_base, 79
  - punnets\_private::tneuron, 76
  - punnets\_private::tneuron\_ext, 81
  - punnets\_private::tneuron\_ext\_const, 84
- getCurrentThrLevel
  - punnets\_common::tneuron\_base, 79
  - punnets\_private::tneuron, 76
  - punnets\_private::tneuron\_ext, 81
- getDeb
  - punnets\_common::debugflag< false >, 31
  - punnets\_common::debugflag< true >, 32
- getDelay

- punnets\_common::tsynapse\_base, 94
- getDescription
  - punnets\_common::func\_base, 34
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 40
  - punnets\_common::func\_exp, 46
  - punnets\_common::func\_exp\_diff, 48
  - punnets\_common::func\_exp\_int, 51
  - punnets\_common::func\_response, 53
  - punnets\_common::func\_sine, 55
  - punnets\_common::func\_sine\_int, 57
  - punnets\_common::func\_sineshot, 59
  - punnets\_common::func\_sineshot\_int, 63
  - punnets\_common::func\_step, 65
- getDest
  - punnets\_common::tsynapse\_base, 94
  - punnets\_private::tsynapse, 90
  - punnets\_private::tsynapse\_addfunc, 92
  - punnets\_private::tsynapse\_fatigue, 96
  - punnets\_private::tsynapse\_message, 98
  - punnets\_private::tsynapse\_messfunc, 100
- getExtInput
  - punnets\_private::tneuron\_ext\_const, 84
- getLastFire
  - punnets\_common::tneuron\_base, 79
  - punnets\_private::tneuron, 76
  - punnets\_private::tneuron\_ext, 81
- getLastSimulate
  - punnets\_common::tneuron\_base, 79
  - punnets\_private::tneuron, 76
  - punnets\_private::tneuron\_ext, 81
- getLastSimulateType
  - punnets\_common::tneuron\_base, 79
  - punnets\_private::tneuron\_ext, 82
- getMaxGradient
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 40
  - punnets\_common::func\_exp, 46
  - punnets\_common::func\_exp\_diff, 48
  - punnets\_common::func\_exp\_int, 51
  - punnets\_common::func\_response, 53
  - punnets\_common::func\_sine, 55
  - punnets\_common::func\_sine\_int, 57
  - punnets\_common::func\_sineshot, 59
  - punnets\_common::func\_sineshot\_int, 63
  - punnets\_common::func\_step, 65
- getMessageId
  - punnets\_common::func\_delta\_int::message\_add\_pulse, 42
  - punnets\_common::func\_deriveq\_base::message\_set\_lambda, 44
  - punnets\_common::func\_deriveq\_base::message\_set\_zero\_point, 45
  - punnets\_common::func\_exp\_diff::message\_add\_event\_time, 50
  - punnets\_common::func\_sineshot::message\_set\_t0, 62
  - punnets\_common::message\_base, 68
- getName
  - punnets\_common::tneuron\_base, 79
- getNextIncontinuity
  - punnets\_common::func\_base, 34
  - punnets\_common::func\_delta\_int, 41
  - punnets\_common::func\_exp, 47
  - punnets\_common::func\_exp\_diff, 49
  - punnets\_common::func\_exp\_int, 52
  - punnets\_common::func\_response, 54
  - punnets\_common::func\_sine, 56
  - punnets\_common::func\_sineshot, 60
  - punnets\_common::func\_step, 66
- getNPulses
  - tobserver, 87
- getSrc
  - punnets\_common::tsynapse\_base, 94
  - punnets\_private::tsynapse, 90
  - punnets\_private::tsynapse\_addfunc, 92
  - punnets\_private::tsynapse\_fatigue, 96
  - punnets\_private::tsynapse\_message, 98
  - punnets\_private::tsynapse\_messfunc, 100
- getSynapses
  - punnets\_private::tneuron, 77
  - punnets\_private::tneuron\_ext, 82
- getTime
  - punnets\_common::tevent, 71
- getValue
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 41
  - punnets\_common::func\_exp, 46
  - punnets\_common::func\_exp\_diff, 48
  - punnets\_common::func\_exp\_int, 51
  - punnets\_common::func\_response, 53
  - punnets\_common::func\_sine, 55
  - punnets\_common::func\_sine\_int, 57
  - punnets\_common::func\_sineshot, 59
  - punnets\_common::func\_sineshot\_int, 63
  - punnets\_common::func\_step, 65
- getValueDomain
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_const, 36
  - punnets\_common::func\_const\_int, 38
  - punnets\_common::func\_delta\_int, 40

- punnets\_common::func\_exp, 46
- punnets\_common::func\_exp\_diff, 48
- punnets\_common::func\_exp\_int, 51
- punnets\_common::func\_response, 53
- punnets\_common::func\_sine, 55
- punnets\_common::func\_sine\_int, 57
- punnets\_common::func\_sineshot, 60
- punnets\_common::func\_sineshot\_int, 63
- punnets\_common::func\_step, 65
- getWeight
  - punnets\_common::tsynapse\_base, 94
  - punnets\_private::tsynapse, 90
  - punnets\_private::tsynapse\_fatigue, 96
- gnuplot\_def
  - punnets\_common::tlogger, 73
- isScheduled
  - punnets\_common::tsched\_double, 88
- lambda
  - punnets\_private::tneuron\_ext, 83
- last\_fire
  - punnets\_private::tneuron\_ext, 83
- last\_simulate
  - punnets\_private::tneuron\_ext, 82
- last\_simulate\_type
  - punnets\_private::tneuron\_ext, 83
- Logging (drawing a graph of neuron potentials), 15
- logoption
  - punnets\_common::tlogger, 73
- loopback
  - punnets\_private::tneuron\_ext, 83
- makePulse
  - punnets\_common, 24
- max\_threshold
  - punnets\_private::tneuron, 77
- min\_threshold
  - punnets\_private::tneuron, 77
- name
  - punnets\_common::tneuron\_base, 80
- ndelay
  - punnets\_common::tsynapse\_base, 94
- Neurons, 18
- ntime\_t
  - punnets\_common, 25
- posstrs
  - dlanguage.cpp, 104
- processMessage
  - punnets\_common::func\_base, 34
  - punnets\_common::func\_delta\_int, 40
  - punnets\_common::func\_deriveq\_base, 43
  - punnets\_common::func\_exp\_diff, 49
  - punnets\_common::func\_sineshot, 61
- pulseArrive
  - punnets\_common::tneuron\_base, 80
  - punnets\_private::tneuron, 76
  - punnets\_private::tneuron\_ext, 81
  - tobserver, 87
- pulses
  - punnets\_private::tneuron\_ext, 82
- punnets, 21
  - tneuron, 21
  - tneuron\_ext, 21
  - tneuron\_ext\_const, 21
  - tsynapse, 21
  - tsynapse\_addfunc, 21
  - tsynapse\_fatigue, 21
  - tsynapse\_message, 21
  - tsynapse\_messfunc, 21
- punnets.h, 114
- punnets\_base.h, 115
- punnets\_common, 23
  - epsilon, 25
  - makePulse, 24
  - ntime\_t, 25
  - real, 27
  - setExtInput, 24
  - totalfiltered\_incontinuity, 25
  - totalfiltered\_maxgrad, 25
  - totalfiltered\_nextpulse, 25
  - totalfire, 24
  - totalpartition, 24
  - totalpeakenclosing, 24
  - totalpulse, 24
  - totalrescheduled, 24
- punnets\_common::debugflag< false >, 31
  - getDeb, 31
  - setDeb, 31
- punnets\_common::debugflag< true >, 32
  - debug, 32
  - debugflag, 32
  - getDeb, 32
  - setDeb, 32
- punnets\_common::func\_base, 33
  - clone, 33
  - get1stDeriv, 33
  - get1stDerivDomain, 33
  - get2ndDeriv, 33
  - get2ndDerivDomain, 33
  - getDescription, 34
  - getMaxGradient, 33
  - getNextIncontinuity, 34
  - getValue, 33
  - getValueDomain, 33
  - processMessage, 34

- setLambda, 33
- setZeroPoint, 33
- shouldDelete, 33
- valueChange, 34
- punnets\_common::func\_const, 36
  - clone, 36
  - func\_const, 36
  - get1stDeriv, 36
  - get1stDerivDomain, 36
  - get2ndDeriv, 36
  - get2ndDerivDomain, 36
  - getDescription, 36
  - getMaxGradient, 36
  - getValue, 36
  - getValueDomain, 36
- punnets\_common::func\_const\_int, 38
  - clone, 38
  - func\_const\_int, 38
  - get1stDeriv, 38
  - get1stDerivDomain, 38
  - get2ndDeriv, 38
  - get2ndDerivDomain, 38
  - getDescription, 38
  - getMaxGradient, 38
  - getValue, 38
  - getValueDomain, 38
- punnets\_common::func\_delta\_int, 40
  - clone, 41
  - func\_delta\_int, 40
  - get1stDeriv, 40
  - get1stDerivDomain, 40
  - get2ndDeriv, 40
  - get2ndDerivDomain, 40
  - getDescription, 40
  - getMaxGradient, 40
  - getNextIncontinuity, 41
  - getValue, 41
  - getValueDomain, 40
  - processMessage, 40
- punnets\_common::func\_delta\_int::message\_-
  - add\_pulse, 42
  - getMessageId, 42
- punnets\_common::func\_deriveq\_base, 43
  - processMessage, 43
  - setLambda, 43
  - setZeroPoint, 43
  - zeropChange, 43
- punnets\_common::func\_deriveq\_-
  - base::message\_set\_lambda, 44
  - getMessageId, 44
- punnets\_common::func\_deriveq\_-
  - base::message\_set\_zero\_point, 45
  - getMessageId, 45
- punnets\_common::func\_exp, 46
  - clone, 46
  - func\_exp, 46
  - get1stDeriv, 46
  - get1stDerivDomain, 46
  - get2ndDeriv, 46
  - get2ndDerivDomain, 46
  - getDescription, 46
  - getMaxGradient, 46
  - getNextIncontinuity, 47
  - getValue, 46
  - getValueDomain, 46
- punnets\_common::func\_exp\_diff, 48
  - clone, 48
  - func\_exp\_diff, 48
  - get1stDeriv, 48
  - get1stDerivDomain, 48
  - get2ndDeriv, 48
  - get2ndDerivDomain, 48
  - getDescription, 48
  - getMaxGradient, 48
  - getNextIncontinuity, 49
  - getValue, 48
  - getValueDomain, 48
  - processMessage, 49
- punnets\_common::func\_exp\_diff::message\_-
  - add\_event\_time, 50
  - getMessageId, 50
- punnets\_common::func\_exp\_int, 51
  - clone, 51
  - func\_exp\_int, 51
  - get1stDeriv, 51
  - get1stDerivDomain, 51
  - get2ndDeriv, 51
  - get2ndDerivDomain, 51
  - getDescription, 51
  - getMaxGradient, 51
  - getNextIncontinuity, 52
  - getValue, 51
  - getValueDomain, 51
- punnets\_common::func\_response, 53
  - clone, 53
  - get1stDeriv, 53
  - get1stDerivDomain, 53
  - get2ndDeriv, 53
  - get2ndDerivDomain, 53
  - getDescription, 53
  - getMaxGradient, 53
  - getNextIncontinuity, 54
  - getValue, 53
  - getValueDomain, 53
  - setZeroPoint, 54
- punnets\_common::func\_sine, 55
  - clone, 55



- func\_sine, 55
- get1stDeriv, 55
- get1stDerivDomain, 55
- get2ndDeriv, 55
- get2ndDerivDomain, 55
- getDescription, 55
- getMaxGradient, 55
- getNextIncontinuity, 56
- getValue, 55
- getValueDomain, 55
- punnets\_common::func\_sine\_int, 57
  - clone, 57
  - func\_sine\_int, 57
  - get1stDeriv, 57
  - get1stDerivDomain, 57
  - get2ndDeriv, 57
  - get2ndDerivDomain, 57
  - getDescription, 57
  - getMaxGradient, 57
  - getValue, 57
  - getValueDomain, 57
- punnets\_common::func\_sineshot, 59
  - clone, 59
  - func\_sineshot, 59
  - get1stDeriv, 59
  - get1stDerivDomain, 59
  - get2ndDeriv, 59
  - get2ndDerivDomain, 59
  - getDescription, 59
  - getMaxGradient, 59
  - getNextIncontinuity, 60
  - getValue, 59
  - getValueDomain, 60
  - processMessage, 61
  - shouldDelete, 59
- punnets\_common::func\_sineshot::message\_-
  - set\_t0, 62
  - getMessageId, 62
- punnets\_common::func\_sineshot\_int, 63
  - clone, 63
  - func\_sineshot\_int, 63
  - get1stDeriv, 63
  - get1stDerivDomain, 63
  - get2ndDeriv, 63
  - get2ndDerivDomain, 63
  - getDescription, 63
  - getMaxGradient, 63
  - getValue, 63
  - getValueDomain, 63
- punnets\_common::func\_step, 65
  - clone, 65
  - get1stDeriv, 65
  - get1stDerivDomain, 65
  - get2ndDeriv, 65
  - get2ndDerivDomain, 65
  - getDescription, 65
  - getMaxGradient, 65
  - getNextIncontinuity, 66
  - getValue, 65
  - getValueDomain, 65
- punnets\_common::greater\_tevent, 67
- punnets\_common::message\_base, 68
  - getMessageId, 68
- punnets\_common::taction, 69
  - activate, 69
  - getClassName, 69
  - queue, 69
- punnets\_common::tevent, 71
  - activate, 71
  - getAction, 71
  - getTime, 71
  - tevent, 71
- punnets\_common::tlogger, 72
  - activate, 72
  - add, 72
  - getClassName, 72
  - gnuplot\_def, 73
  - logoption, 73
  - queue, 72
  - schedule, 72
  - showext, 73
  - shownone, 73
  - showpart, 73
  - showthr, 73
  - tlogger, 72
- punnets\_common::tneuron\_base, 79
  - \_queue, 80
  - addSynapse, 79
  - getClassName, 80
  - getCurrentExtInput, 79
  - getCurrentSigLevel, 79
  - getCurrentThrLevel, 79
  - getLastFire, 79
  - getLastSimulate, 79
  - getLastSimulateType, 79
  - getName, 79
  - name, 80
  - pulseArrive, 80
  - queue, 79
  - tneuron\_base, 79
- punnets\_common::tsched\_double, 88
  - isScheduled, 88
  - run, 88
  - scheduleEvent, 88
  - tsched\_double, 88
- punnets\_common::tsentinel, 89
  - activate, 89
  - getClassName, 89

- queue, 89
- punnets\_common::tsynapse\_base, 94
  - activate, 95
  - getClassName, 94
  - getDelay, 94
  - getDest, 94
  - getSrc, 94
  - getWeight, 94
  - ndelay, 94
  - setSrc, 94
  - tsynapse\_base, 94
- punnets\_nodebug, 29
  - tneuron, 29
  - tneuron\_ext, 29
  - tneuron\_ext\_const, 29
  - tsynapse, 29
  - tsynapse\_addfunc, 29
  - tsynapse\_fatigue, 29
  - tsynapse\_message, 29
  - tsynapse\_messfunc, 29
- punnets\_private, 30
- punnets\_private::tneuron, 76
  - activate, 76
  - addSynapse, 76
  - coeff\_sigdecay, 77
  - coeff\_thrdecay, 77
  - eraseSynapse, 76
  - getClassName, 76
  - getCurrentSigLevel, 76
  - getCurrentThrLevel, 76
  - getLastFire, 76
  - getLastSimulate, 76
  - getSynapses, 77
  - max\_threshold, 77
  - min\_threshold, 77
  - pulseArrive, 76
  - queue, 76
  - scheduleFire, 77
  - sig\_converge\_level, 77
  - sig\_hv\_period, 77
  - simulateElapse, 77
  - thr\_hv\_period, 77
  - tneuron, 78
- punnets\_private::tneuron\_ext, 81
  - activate, 82
  - addExt, 81
  - addSynapse, 81
  - broadcastMessage, 82
  - calcSignal, 82
  - eraseSynapse, 81
  - exts, 82
  - fire, 82
  - getClassName, 82
  - getCurrentSigLevel, 81
  - getCurrentThrLevel, 81
  - getLastFire, 81
  - getLastSimulate, 81
  - getLastSimulateType, 82
  - getSynapses, 82
  - lambda, 83
  - last\_fire, 83
  - last\_simulate, 82
  - last\_simulate\_type, 83
  - loopback, 83
  - pulseArrive, 81
  - pulses, 82
  - queue, 82
  - scheduleFire, 82
  - sendMessage, 82
  - setLoopBack, 83
  - synapses, 82
  - tneuron\_ext, 81
- punnets\_private::tneuron\_ext\_const, 84
  - getClassName, 84
  - getConvergeLevel, 84
  - getCurrentExtInput, 84
  - getCurrentSigLevel, 84
  - getExtInput, 84
  - scheduleFire, 84
  - setConvergeLevel, 85
  - setExtInput, 84
  - sig\_converge\_level, 85
  - simulateElapse, 85
  - tneuron\_ext\_const, 85
- punnets\_private::tsynapse, 90
  - activate, 90
  - addDelay, 90
  - addWeight, 90
  - getDest, 90
  - getSrc, 90
  - getWeight, 90
  - queue, 90
  - setSrc, 90
  - tsynapse, 90, 91
- punnets\_private::tsynapse\_addfunc, 92
  - activate, 92
  - getDest, 92
  - getSrc, 92
  - queue, 92
  - setSrc, 92
  - tsynapse\_addfunc, 93
- punnets\_private::tsynapse\_fatigue, 96
  - activate, 97
  - addDelay, 96
  - addWeight, 96
  - fire\_ratio, 96
  - getClassName, 96
  - getDest, 96

- getSrc, 96
- getWeight, 96
- recover\_hv\_period, 96
- tsynapse\_fatigue, 96, 97
- punnets\_private::tsynapse\_message, 98
- ~tsynapse\_message, 98
- activate, 98
- getDest, 98
- getSrc, 98
- setSrc, 98
- tsynapse\_message, 99
- punnets\_private::tsynapse\_messfunc, 100
- activate, 101
- getDest, 100
- getSrc, 100
- queue, 100
- setSrc, 100
- tsynapse\_messfunc, 101
- queue
  - punnets\_common::taction, 69
  - punnets\_common::tlogger, 72
  - punnets\_common::tneuron\_base, 79
  - punnets\_common::tsentinel, 89
  - punnets\_private::tneuron, 76
  - punnets\_private::tneuron\_ext, 82
  - punnets\_private::tsynapse, 90
  - punnets\_private::tsynapse\_addfunc, 92
  - punnets\_private::tsynapse\_messfunc, 100
- real
  - punnets\_common, 27
- recover\_hv\_period
  - punnets\_private::tsynapse\_fatigue, 96
- run
  - punnets\_common::tsched\_double, 88
- schedule
  - punnets\_common::tlogger, 72
- scheduleEvent
  - punnets\_common::tsched\_double, 88
- scheduleFire
  - punnets\_private::tneuron, 77
  - punnets\_private::tneuron\_ext, 82
  - punnets\_private::tneuron\_ext\_const, 84
- Scheduling, 20
- sendMessage
  - punnets\_private::tneuron\_ext, 82
- setConvergeLevel
  - punnets\_private::tneuron\_ext\_const, 85
- setDeb
  - punnets\_common::debugflag< false >, 31
- punnets\_common::debugflag< true >, 32
- setExtInput
  - punnets\_common, 24
  - punnets\_private::tneuron\_ext\_const, 84
- setLambda
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_deriveq\_base, 43
- setLoopBack
  - punnets\_private::tneuron\_ext, 83
- setSrc
  - punnets\_common::tsynapse\_base, 94
  - punnets\_private::tsynapse, 90
  - punnets\_private::tsynapse\_addfunc, 92
  - punnets\_private::tsynapse\_message, 98
  - punnets\_private::tsynapse\_messfunc, 100
- setZeroPoint
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_deriveq\_base, 43
  - punnets\_common::func\_response, 54
- shouldDelete
  - punnets\_common::func\_base, 33
  - punnets\_common::func\_sineshot, 59
- showext
  - punnets\_common::tlogger, 73
- shownone
  - punnets\_common::tlogger, 73
- showpart
  - punnets\_common::tlogger, 73
- showthr
  - punnets\_common::tlogger, 73
- sig\_converge\_level
  - punnets\_private::tneuron, 77
  - punnets\_private::tneuron\_ext\_const, 85
- sig\_hv\_period
  - punnets\_private::tneuron, 77
- simulateElapse
  - punnets\_private::tneuron, 77
  - punnets\_private::tneuron\_ext\_const, 85
- Synapses, 19
- synapses
  - punnets\_private::tneuron\_ext, 82
- tevent
  - punnets\_common::tevent, 71
- thr\_hv\_period
  - punnets\_private::tneuron, 77
- tlogger
  - punnets\_common::tlogger, 72
- tneuron
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tneuron, 78
- tneuron\_base

- punnets\_common::tneuron\_base, 79
- tneuron\_ext
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tneuron\_ext, 81
- tneuron\_ext\_const
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tneuron\_ext\_const, 85
- tobserver, 87
  - clearNPulses, 87
  - getNPulses, 87
  - pulseArrive, 87
  - tobserver, 87
- totalfiltered\_incontinuity
  - punnets\_common, 25
  - totalvars, 16
- totalfiltered\_maxgrad
  - punnets\_common, 25
  - totalvars, 16
- totalfiltered\_nextpulse
  - punnets\_common, 25
  - totalvars, 16
- totalfire
  - punnets\_common, 24
  - totalvars, 16
- totalpartition
  - punnets\_common, 24
  - totalvars, 16
- totalpartition\_newton
  - totalvars, 16
- totalpartition\_nonewton
  - totalvars, 16
- totalpeakenclosing
  - punnets\_common, 24
  - totalvars, 16
- totalpeaksearch
  - totalvars, 16
- totalpulse
  - punnets\_common, 24
  - totalvars, 16
- totalrescheduled
  - punnets\_common, 24
  - totalvars, 16
- totalvars
  - totalfiltered\_incontinuity, 16
  - totalfiltered\_maxgrad, 16
  - totalfiltered\_nextpulse, 16
  - totalfire, 16
  - totalpartition, 16
  - totalpartition\_newton, 16
  - totalpartition\_nonewton, 16
  - totalpeakenclosing, 16
  - totalpeaksearch, 16
  - totalpulse, 16
  - totalrescheduled, 16
- tsched\_double
  - punnets\_common::tsched\_double, 88
- tsynapse
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tsynapse, 90, 91
- tsynapse\_addfunc
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tsynapse\_addfunc, 93
- tsynapse\_base
  - punnets\_common::tsynapse\_base, 94
- tsynapse\_fatigue
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tsynapse\_fatigue, 96, 97
- tsynapse\_message
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tsynapse\_message, 99
- tsynapse\_messfunc
  - punnets, 21
  - punnets\_nodebug, 29
  - punnets\_private::tsynapse\_messfunc, 101
- valueChange
  - punnets\_common::func\_base, 34
- Variables for Statistics, 16
- zeropChange
  - punnets\_common::func\_deriveq\_base, 43