

HPWHsim

1.2.1

Generated by Doxygen 1.8.6

Wed Mar 30 2016 14:43:50



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List	1
<b>2</b>	<b>Class Documentation</b>	<b>3</b>
2.1	HPWH::HeatSource Class Reference	3
2.1.1	Constructor & Destructor Documentation	3
2.1.1.1	HeatSource	3
2.1.2	Member Function Documentation	3
2.1.2.1	addHeat	3
2.1.2.2	disengageHeatSource	4
2.1.2.3	engageHeatSource	4
2.1.2.4	isEngaged	4
2.1.2.5	setCondensity	4
2.1.2.6	setUpAsResistiveElement	4
2.1.2.7	shouldHeat	4
2.1.2.8	shutsOff	4
2.2	HPWH Class Reference	4
2.2.1	Detailed Description	6
2.2.2	Member Enumeration Documentation	6
2.2.2.1	DRMODES	6
2.2.2.2	HEATSOURCE_TYPE	7
2.2.2.3	MODELS	7
2.2.2.4	UNITS	7
2.2.2.5	VERBOSITY	8
2.2.3	Constructor & Destructor Documentation	8
2.2.3.1	HPWH	8
2.2.3.2	HPWH	8
2.2.3.3	~HPWH	8
2.2.4	Member Function Documentation	8
2.2.4.1	getEnergyRemovedFromEnvironment	8
2.2.4.2	getNthHeatSourceEnergyInput	8

2.2.4.3	getNthHeatSourceEnergyInput	8
2.2.4.4	getNthHeatSourceEnergyOutput	8
2.2.4.5	getNthHeatSourceEnergyOutput	9
2.2.4.6	getNthHeatSourceRunTime	9
2.2.4.7	getNthHeatSourceType	9
2.2.4.8	getNthSimTcouple	9
2.2.4.9	getNthSimTcouple	9
2.2.4.10	getNumHeatSources	9
2.2.4.11	getNumNodes	9
2.2.4.12	getOutletTemp	9
2.2.4.13	getSetpoint	9
2.2.4.14	getStandbyLosses	10
2.2.4.15	getTankNodeTemp	10
2.2.4.16	getTankNodeTemp	10
2.2.4.17	getVersion	10
2.2.4.18	HPWHinit_file	10
2.2.4.19	HPWHinit_presets	10
2.2.4.20	HPWHinit_resTank	10
2.2.4.21	HPWHinit_resTank	10
2.2.4.22	isNthHeatSourceRunning	11
2.2.4.23	isSetpointFixed	11
2.2.4.24	operator=	11
2.2.4.25	printHeatSourceInfo	11
2.2.4.26	printTankTemps	11
2.2.4.27	resetTankToSetpoint	11
2.2.4.28	runNSteps	11
2.2.4.29	runOneStep	11
2.2.4.30	runOneStep	11
2.2.4.31	setAirFlowFreedom	11
2.2.4.32	setDoTempDepression	12
2.2.4.33	setInletT	12
2.2.4.34	setMessageCallback	12
2.2.4.35	setSetpoint	12
2.2.4.36	setSetpoint	12
2.2.4.37	setTankSize	12
2.2.4.38	setUA	12
2.2.4.39	setVerbosity	12
2.2.4.40	WriteCSVRow	12
2.2.5	Member Data Documentation	12
2.2.5.1	CONDENSITY_SIZE	12

---

2.2.5.2	HEATDIST_MINVALUE . . . . .	12
2.2.5.3	HPWH_ABORT . . . . .	13
2.2.5.4	MAXOUTSTRING . . . . .	13
2.2.5.5	UNINITIALIZED_LOCATIONTEMP . . . . .	13
<b>Index</b>		<b>14</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">HPWH::HeatSource</a> . . . . .	<a href="#">3</a>
<a href="#">HPWH</a> . . . . .	<a href="#">4</a>





## Chapter 2

# Class Documentation

## 2.1 HPWH::HeatSource Class Reference

### Public Member Functions

- [HeatSource](#) ([HPWH](#) \*parentHPWH)
- **HeatSource** (const [HeatSource](#) &hSource)
- [HeatSource](#) & [operator=](#) (const [HeatSource](#) &hSource)  
*copy constructor*
- void [setupAsResistiveElement](#) (int node, double Watts)  
*assignment operator*
- bool [isEngaged](#) () const
- void [engageHeatSource](#) (double heatSourceAmbientT\_C)
- void [disengageHeatSource](#) ()
- bool [shouldHeat](#) (double heatSourceAmbientT\_C) const
- bool [shutsOff](#) (double heatSourceAmbientT\_C) const
- void **addHeat\_temp** (double externalT\_C, double minutesPerStep)
- void [addHeat](#) (double externalT\_C, double minutesToRun)
- void [setCondensity](#) (double cnd1, double cnd2, double cnd3, double cnd4, double cnd5, double cnd6, double cnd7, double cnd8, double cnd9, double cnd10, double cnd11, double cnd12)

### Friends

- class **HPWH**

### 2.1.1 Constructor & Destructor Documentation

#### 2.1.1.1 HPWH::HeatSource::HeatSource ( [HPWH](#) \* *parentHPWH* )

default constructor, does not create a useful [HeatSource](#) constructor assigns a pointer to the hpwh that owns this heat source

### 2.1.2 Member Function Documentation

#### 2.1.2.1 void HPWH::HeatSource::addHeat ( double *externalT\_C*, double *minutesToRun* )

adds heat to the hpwh - this is the function that interprets the various configurations (internal/external, resistance/heat pump) to add heat

2.1.2.2 void HPWH::HeatSource::disengageHeatSource ( )

turn heat source off, i.e. set isEngaged to FALSE

2.1.2.3 void HPWH::HeatSource::engageHeatSource ( double *heatSourceAmbientT\_C* )

turn heat source on, i.e. set isEngaged to TRUE

2.1.2.4 bool HPWH::HeatSource::isEngaged ( ) const

return whether or not the heat source is engaged

2.1.2.5 void HPWH::HeatSource::setCondensity ( double *cnd1*, double *cnd2*, double *cnd3*, double *cnd4*, double *cnd5*, double *cnd6*, double *cnd7*, double *cnd8*, double *cnd9*, double *cnd10*, double *cnd11*, double *cnd12* )

a function to set the condensity values, it pretties up the init funcs.

2.1.2.6 void HPWH::HeatSource::setupAsResistiveElement ( int *node*, double *Watts* )

assignment operator

< the copy constructor and assignment operator basically just checks if there are backup/companion pointers - these can't be copied configure the heat source to be a resistive element, positioned at the specified node, with the specified power in watts

2.1.2.7 bool HPWH::HeatSource::shouldHeat ( double *heatSourceAmbientT\_C* ) const

queries the heat source as to whether or not it should turn on

2.1.2.8 bool HPWH::HeatSource::shutsOff ( double *heatSourceAmbientT\_C* ) const

queries the heat source whether should shut off (typically lowT shutoff)

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

- /storage/server/nkvaltine/Projects/HPWHsim/dev/HPWH.hh
- /storage/server/nkvaltine/Projects/HPWHsim/dev/HPWH.cc

## 2.2 HPWH Class Reference

```
#include <HPWH.hh>
```

### Classes

- class [HeatSource](#)

### Public Types

- enum [DRMODES](#) { [DR\\_BLOCK](#) = 0, [DR\\_ALLOW](#) = 1, [DR\\_ENGAGE](#) = 2 }

- enum `MODELS` {  
`MODELS_restankNoUA` = 1, `MODELS_restankHugeUA` = 2, `MODELS_restankRealistic` = 3, `MODELS_basicIntegrated` = 4,  
`MODELS_externalTest` = 5, `MODELS_AOSmithPHPT60` = 102, `MODELS_AOSmithPHPT80` = 103, `MODELS_AOSmithHPTU50` = 104,  
`MODELS_AOSmithHPTU66` = 105, `MODELS_AOSmithHPTU80` = 106, `MODELS_GE2012` = 110, `MODELS_GE2014STDMode` = 111,  
`MODELS_GE2014` = 112, `MODELS_Sanden40` = 120, `MODELS_Sanden80` = 121, `MODELS_RheemHB50` = 140,  
`MODELS_Stiebel220E` = 150, `MODELS_Generic1` = 160, `MODELS_Generic2` = 161, `MODELS_Generic3` = 162,  
`MODELS_CustomFile` = 200, `MODELS_CustomResTank` = 201 }
- enum `VERBOSITY` {  
`VRB_silent` = 0, `VRB_reluctant` = 10, `VRB_minuteOut` = 15, `VRB_typical` = 20,  
`VRB_emetic` = 30 }
- enum `UNITS` {  
`UNITS_C`, `UNITS_F`, `UNITS_KWH`, `UNITS_BTU`,  
`UNITS_KJ`, `UNITS_GAL`, `UNITS_L`, `UNITS_kJperHrC`,  
`UNITS_BTUperHrF` }
- enum `HEATSOURCE_TYPE` { `TYPE_none`, `TYPE_resistance`, `TYPE_compressor` }

## Public Member Functions

- `HPWH` ()
- `HPWH` (const `HPWH` &hpwh)
- `HPWH` & `operator=` (const `HPWH` &hpwh)
- `~HPWH` ()
- int `HPWHinit_presets` (`MODELS` presetNum)
- int `HPWHinit_file` (std::string configFile)
- int `HPWHinit_resTank` ()
- int `HPWHinit_resTank` (double tankVol\_L, double energyFactor, double upperPower\_W, double lowerPower\_W)
- int `runOneStep` (double inletT\_C, double drawVolume\_L, double ambientT\_C, double externalT\_C, `DRMODES` DRstatus, double minutesPerStep)
- int `runNSteps` (int N, double \*inletT\_C, double \*drawVolume\_L, double \*tankAmbientT\_C, double \*heatSourceAmbientT\_C, `DRMODES` \*DRstatus, double minutesPerStep)
- void `setVerbosity` (`VERBOSITY` hpwhVrb)
- void `setMessageCallback` (void(\*callbackFunc)(const std::string message, void \*pContext), void \*pContext)
- void `printHeatSourceInfo` ()
- void `printTankTemps` ()
- int `WriteCSVHeading` (FILE \*outFILE, const char \*preamble="") const
- int `WriteCSVRow` (FILE \*outFILE, const char \*preamble="") const
- bool `isSetpointFixed` ()
- int `setSetpoint` (double newSetpoint)
- int `setSetpoint` (double newSetpoint, `UNITS` units)
- double `getSetpoint` ()
- int `resetTankToSetpoint` ()
- int `setAirFlowFreedom` (double fanFraction)
- int `setDoTempDepression` (bool doTempDepress)
- int `setTankSize` (double HPWH\_size\_L)
- int `setTankSize` (double HPWH\_size, `UNITS` units)
- int `setUA` (double UA\_kJperHrC)
- int `setUA` (double UA, `UNITS` units)
- int `getNumNodes` () const
- double `getTankNodeTemp` (int nodeNum) const

- double `getTankNodeTemp` (int nodeNum, `UNITS` units) const
- double `getNthSimTcouple` (int N) const
- double `getNthSimTcouple` (int N, `UNITS` units) const
- int `getNumHeatSources` () const
- double `getNthHeatSourceEnergyInput` (int N) const
- double `getNthHeatSourceEnergyInput` (int N, `UNITS` units) const
- double `getNthHeatSourceEnergyOutput` (int N) const
- double `getNthHeatSourceEnergyOutput` (int N, `UNITS` units) const
- double `getNthHeatSourceRunTime` (int N) const
- int `isNthHeatSourceRunning` (int N) const
- `HEATSOURCE_TYPE` `getNthHeatSourceType` (int N) const
- double `getOutletTemp` () const
- double `getOutletTemp` (`UNITS` units) const
- double `getEnergyRemovedFromEnvironment` () const
- double `getEnergyRemovedFromEnvironment` (`UNITS` units) const
- double `getStandbyLosses` () const
- double `getStandbyLosses` (`UNITS` units) const
- int `runOneStep` (double drawVolume\_L, double ambientT\_C, double externalT\_C, `DRMODES` DRstatus)
- void `setInletT` (double newInletT\_C)
- void `setMinutesPerStep` (double newMinutesPerStep)

## Static Public Member Functions

- static std::string `getVersion` ()

## Static Public Attributes

- static const int `version_major` = 1
- static const int `version_minor` = 2
- static const int `version_maint` = 1
- static const float `DENSITYWATER_kgperL` = 0.998f
- static const float `CPWATER_kJperkgC` = 4.181f
- static const int `CONDENSITY_SIZE` = 12
- static const int `MAXOUTSTRING` = 200
- static const float `HEATDIST_MINVALUE` = 0.0001f
- static const float `UNINITIALIZED_LOCATIONTEMP` = -500.f
- static const int `HPWH_ABORT` = -274000

### 2.2.1 Detailed Description

< If `HPWH_ABRIDGED` is defined, then some function definitions will be excluded from compiling. This is done in order to reduce the size of the final compiled code.

### 2.2.2 Member Enumeration Documentation

#### 2.2.2.1 enum `HPWH::DRMODES`

specifies the various modes for the Demand Response (DR) abilities values may vary - names should be used

Enumerator

**`DR_BLOCK`** this mode prohibits the elements from engaging and turns off any currently running

**`DR_ALLOW`** this mode allows the water heater to run normally

**`DR_ENGAGE`** this mode forces an element to turn on

## 2.2.2.2 enum HPWH::HEATSOURCE\_TYPE

specifies the type of heat source

Enumerator

**TYPE\_none** a default to check to make sure it's been set

**TYPE\_resistance** a resistance element

**TYPE\_compressor** a vapor cycle compressor

## 2.2.2.3 enum HPWH::MODELS

specifies the allowable preset [HPWH](#) models values may vary - names should be used

Enumerator

**MODELS\_restankNoUA** a simple resistance tank, but with no tank losses

**MODELS\_restankHugeUA** a simple resistance tank, but with very large tank losses

**MODELS\_restankRealistic** a more-or-less realistic resistance tank

**MODELS\_basicIntegrated** a standard integrated [HPWH](#)

**MODELS\_externalTest** a single compressor tank, using "external" topology

**MODELS\_AOSmithPHPT60** this is the Ecotope model for the 60 gallon Voltex [HPWH](#)

**MODELS\_AOSmithPHPT80** Voltex 80 gallon tank

**MODELS\_AOSmithHPTU50** 50 gallon AOSmith HPTU

**MODELS\_AOSmithHPTU66** 66 gallon AOSmith HPTU

**MODELS\_AOSmithHPTU80** 80 gallon AOSmith HPTU

**MODELS\_GE2012** The 2012 era GeoSpring

**MODELS\_GE2014STDMode** 2014 GE model run in standard mode

**MODELS\_GE2014** 2014 GE model run in the efficiency mode

**MODELS\_Sanden40** Sanden 40 gallon CO2 external heat pump

**MODELS\_Sanden80** Sanden 80 gallon CO2 external heat pump

**MODELS\_RheemHB50** Rheem 2014 (?) Model

**MODELS\_Stiebel220E** Stiebel Eltron (2014 model?)

**MODELS\_Generic1** Generic Tier 1

**MODELS\_Generic2** Generic Tier 2

**MODELS\_Generic3** Generic Tier 3

**MODELS\_CustomFile** [HPWH](#) parameters were input via file

**MODELS\_CustomResTank** [HPWH](#) parameters were input via HPWHinit\_resTank

## 2.2.2.4 enum HPWH::UNITS

Enumerator

**UNITS\_C** celsius

**UNITS\_F** fahrenheit

**UNITS\_KWH** kilowatt hours

**UNITS\_BTU** british thermal units

**UNITS\_KJ** kilojoules

**UNITS\_GAL** gallons

**UNITS\_L** liters

**UNITS\_kJperHrC** UA, metric units

**UNITS\_BTUperHrF** UA, imperial units

### 2.2.2.5 enum HPWH::VERBOSITY

specifies the modes for writing output the specified values are used for  $\geq$  comparisons, so the numerical order is relevant

Enumerator

***VRB\_silent*** print no outputs  
***VRB\_reluctant*** print only outputs for fatal errors  
***VRB\_minuteOut*** print minutely output  
***VRB\_typical*** print some basic debugging info  
***VRB\_emetic*** print all the things

## 2.2.3 Constructor & Destructor Documentation

### 2.2.3.1 HPWH::HPWH ( )

default constructor

### 2.2.3.2 HPWH::HPWH ( const HPWH & *hpwh* )

copy constructor

### 2.2.3.3 HPWH::~~HPWH ( )

destructor just a couple dynamic arrays to destroy - could be replaced by vectors eventually?

## 2.2.4 Member Function Documentation

### 2.2.4.1 double HPWH::getEnergyRemovedFromEnvironment ( UNITS *units* ) const

get the total energy removed from the environment by all heat sources in specified units (not net energy - does not include standby) moving heat from the space to the water is the positive direction returns HPWH\_ABORT for incorrect units

### 2.2.4.2 double HPWH::getNthHeatSourceEnergyInput ( int *N* ) const

Parameters

<i>N</i>	default units kWh
----------	-------------------

### 2.2.4.3 double HPWH::getNthHeatSourceEnergyInput ( int *N*, UNITS *units* ) const

returns the energy input to the Nth heat source, with the specified units energy used by the heat source is positive - should always be positive returns HPWH\_ABORT for N out of bounds or incorrect units

### 2.2.4.4 double HPWH::getNthHeatSourceEnergyOutput ( int *N* ) const

## Parameters

<i>N</i>	default units kWh
----------	-------------------

**2.2.4.5** double HPWH::getNthHeatSourceEnergyOutput ( int *N*, **UNITS** *units* ) const

returns the energy output from the Nth heat source, with the specified units energy put into the water is positive - should always be positive returns HPWH\_ABORT for N out of bounds or incorrect units

**2.2.4.6** double HPWH::getNthHeatSourceRunTime ( int *N* ) const

returns the run time for the Nth heat source, in minutes note: they may sum to more than 1 time step for concurrently running heat sources returns HPWH\_ABORT for N out of bounds

**2.2.4.7** HPWH::HEATSOURCE\_TYPE HPWH::getNthHeatSourceType ( int *N* ) const

returns the enum value for what type of heat source the Nth heat source is

**2.2.4.8** double HPWH::getNthSimTcouple ( int *N* ) const

## Parameters

<i>N</i>	default units C
----------	-----------------

**2.2.4.9** double HPWH::getNthSimTcouple ( int *N*, **UNITS** *units* ) const

returns the temperature from a set of 6 virtual "thermocouples", which are constructed from the node temperature array. Specify t-couple from 1-6, 1 at the bottom using specified units returns HPWH\_ABORT for N < 0, > 6, or incorrect units

**2.2.4.10** int HPWH::getNumHeatSources ( ) const

returns the number of heat sources

**2.2.4.11** int HPWH::getNumNodes ( ) const

returns the number of nodes

**2.2.4.12** double HPWH::getOutletTemp ( **UNITS** *units* ) const

returns the outlet temperature in the specified units returns 0 when no draw occurs, or HPWH\_ABORT for incorrect unit specifier

**2.2.4.13** double HPWH::getSetpoint ( )

a function to check the setpoint - returns setpoint in celcius

#### 2.2.4.14 double HPWH::getStandbyLosses ( UNITS units ) const

get the amount of heat lost through the tank in specified units moving heat from the water to the space is the positive direction negative should occur seldom returns HPWH\_ABORT for incorrect units

#### 2.2.4.15 double HPWH::getTankNodeTemp ( int nodeNum ) const

##### Parameters

<i>nodeNum</i>	default units C
----------------	-----------------

#### 2.2.4.16 double HPWH::getTankNodeTemp ( int nodeNum, UNITS units ) const

returns the temperature of the water at the specified node - with specified units or HPWH\_ABORT for incorrect node number or unit failure

#### 2.2.4.17 string HPWH::getVersion ( ) [static]

This function returns a string with the current version number

#### 2.2.4.18 int HPWH::HPWHinit\_file ( std::string configFile )

This function will load in a set of parameters from a file The file name is the input - there should be at most one set of parameters per file This is useful for testing new variations, and for the sort of variability that we typically do when creating SEEM runs Appropriate use of this function can be found in the documentation

The return value is 0 for successful initialization, HPWH\_ABORT otherwise

#### 2.2.4.19 int HPWH::HPWHinit\_presets ( MODELS presetNum )

This function will load in a set of parameters that are hardcoded in this function - which particular set of parameters is selected by presetNum. This is similar to the way the HPWHsim currently operates, as used in SEEM, but not quite as versatile. My impression is that this could be a useful input paradigm for CSE

The return value is 0 for successful initialization, HPWH\_ABORT otherwise

#### 2.2.4.20 int HPWH::HPWHinit\_resTank ( )

Default resistance tank, EF 0.95, volume 47.5

#### 2.2.4.21 int HPWH::HPWHinit\_resTank ( double tankVol\_L, double energyFactor, double upperPower\_W, double lowerPower\_W )

This function will initialize a [HPWH](#) object to be a resistance tank. Since resistance tanks are so simple, they can be specified with only four variables: tank volume, energy factor, and the power of the upper and lower elements. Energy factor is converted into UA internally, although an external setter for UA is also provided in case the energy factor is unknown.

Several assumptions regarding the tank configuration are assumed: the lower element is at the bottom, the upper element is at the top third. The logics are also set to standard setting, with upper as VIP activating when the top third is too cold.



**2.2.4.22** `int HPWH::isNthHeatSourceRunning ( int N ) const`

returns 1 if the Nth heat source is currently engaged, 0 if it is not, and returns HPWH\_ABORT for N out of bounds

**2.2.4.23** `bool HPWH::isSetpointFixed ( )`

is the setpoint allowed to be changed

**2.2.4.24** `HPWH & HPWH::operator= ( const HPWH & hpwh )`

assignment operator

**2.2.4.25** `void HPWH::printHeatSourceInfo ( )`

this prints out the heat source info, nicely formatted specifically input/output energy/power, and runtime will print to cout if messageCallback pointer is unspecified does not use verbosity, as it is public and expected to be called only when needed

**2.2.4.26** `void HPWH::printTankTemps ( )`

this prints out all the node temps, kind of nicely formatted does not use verbosity, as it is public and expected to be called only when needed

**2.2.4.27** `int HPWH::resetTankToSetpoint ( )`

this function resets the tank temperature profile to be completely at setpoint The return value is 0 for successful completion

**2.2.4.28** `int HPWH::runNSteps ( int N, double * inletT_C, double * drawVolume_L, double * tankAmbientT_C, double * heatSourceAmbientT_C, DRMODES * DRstatus, double minutesPerStep )`

This function will progress the simulation forward in time by N (equal) steps The calculated values will be summed or averaged, as appropriate, and then stored in the usual variables to be accessed through functions

The return value is 0 for successful simulation run, HPWH\_ABORT otherwise

**2.2.4.29** `int HPWH::runOneStep ( double inletT_C, double drawVolume_L, double ambientT_C, double externalT_C, DRMODES DRstatus, double minutesPerStep )`

This function will progress the simulation forward in time by one step all calculated outputs are stored in private variables and accessed through functions

The return value is 0 for successful simulation run, HPWH\_ABORT otherwise

**2.2.4.30** `int HPWH::runOneStep ( double drawVolume_L, double ambientT_C, double externalT_C, DRMODES DRstatus )  
[inline]`

An overloaded function that uses some member variables, instead of taking them as inputs

**2.2.4.31** `int HPWH::setAirFlowFreedom ( double fanFraction )`

This is a simple setter for the AirFlowFreedom

**2.2.4.32** `int HPWH::setDoTempDepression ( bool doTempDepress )`

This is a simple setter for the temperature depression option

**2.2.4.33** `void HPWH::setInletT ( double newInletT_C ) [inline]`

Setters for the what are typically input variables

**2.2.4.34** `void HPWH::setMessageCallback ( void (*)(const std::string message, void *pContext) callbackFunc, void * pContext )`

sets the function to be used for message passing

**2.2.4.35** `int HPWH::setSetpoint ( double newSetpoint )`

Parameters

<i>newSetpoint</i>	default units C
--------------------	-----------------

**2.2.4.36** `int HPWH::setSetpoint ( double newSetpoint, UNITS units )`

a function to change the setpoint - useful for dynamically setting it The return value is 0 for successful setting, HPWH\_ABORT for units failure

**2.2.4.37** `int HPWH::setTankSize ( double HPWH_size, UNITS units )`

This is a simple setter for the tank volume in L or GAL

**2.2.4.38** `int HPWH::setUA ( double UA, UNITS units )`

This is a setter for the UA, with or without units specified - default is metric

**2.2.4.39** `void HPWH::setVerbosity ( VERBOSITY hpwhVrb )`

sets the verbosity to the specified level

**2.2.4.40** `int HPWH::WriteCSVRow ( FILE * outFILE, const char * preamble = " " ) const`

a couple of function to write the outputs to a file they both will return 0 for success the preamble should be supplied with a trailing comma, as these functions do not add one. Additionally, a newline is written with each call.

## 2.2.5 Member Data Documentation

**2.2.5.1** `const int HPWH::CONDENSITY_SIZE = 12 [static]`

this must be an integer, and only the value 12 change at your own risk

**2.2.5.2** `const float HPWH::HEATDIST_MINVALUE = 0.0001f [static]`

any amount of heat distribution less than this is reduced to 0 this saves on computations

**2.2.5.3** `const int HPWH::HPWH_ABORT = -274000` `[static]`

this is the value that the public functions will return in case of a simulation destroying error

**2.2.5.4** `const int HPWH::MAXOUTSTRING = 200` `[static]`

this is the maximum length for a debugging output string

**2.2.5.5** `const float HPWH::UNINITIALIZED_LOCATIONTEMP = -500.f` `[static]`

this is used to tell the simulation when the location temperature has not been initialized

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

- `/storage/server/nkvaltine/Projects/HPWHsim/dev/HPWH.hh`
- `/storage/server/nkvaltine/Projects/HPWHsim/dev/HPWH.cc`

# Index

~HPWH  
    HPWH, [8](#)

addHeat  
    HPWH::HeatSource, [3](#)

CONDENSITY\_SIZE  
    HPWH, [12](#)

DR\_ALLOW  
    HPWH, [6](#)

DR\_BLOCK  
    HPWH, [6](#)

DR\_ENGAGE  
    HPWH, [6](#)

DRMODES  
    HPWH, [6](#)

disengageHeatSource  
    HPWH::HeatSource, [3](#)

engageHeatSource  
    HPWH::HeatSource, [4](#)

getEnergyRemovedFromEnvironment  
    HPWH, [8](#)

getNthHeatSourceEnergyInput  
    HPWH, [8](#)

getNthHeatSourceEnergyOutput  
    HPWH, [8](#), [9](#)

getNthHeatSourceRunTime  
    HPWH, [9](#)

getNthHeatSourceType  
    HPWH, [9](#)

getNthSimTcouple  
    HPWH, [9](#)

getNumHeatSources  
    HPWH, [9](#)

getNumNodes  
    HPWH, [9](#)

getOutletTemp  
    HPWH, [9](#)

getSetpoint  
    HPWH, [9](#)

getStandbyLosses  
    HPWH, [9](#)

getTankNodeTemp  
    HPWH, [10](#)

getVersion  
    HPWH, [10](#)

HPWH

DR\_ALLOW, [6](#)

DR\_BLOCK, [6](#)

DR\_ENGAGE, [6](#)

MODELS\_AOSmithHPTU50, [7](#)

MODELS\_AOSmithHPTU66, [7](#)

MODELS\_AOSmithHPTU80, [7](#)

MODELS\_AOSmithPHPT60, [7](#)

MODELS\_AOSmithPHPT80, [7](#)

MODELS\_CustomFile, [7](#)

MODELS\_CustomResTank, [7](#)

MODELS\_GE2012, [7](#)

MODELS\_GE2014, [7](#)

MODELS\_GE2014STDMode, [7](#)

MODELS\_Generic1, [7](#)

MODELS\_Generic2, [7](#)

MODELS\_Generic3, [7](#)

MODELS\_RheemHB50, [7](#)

MODELS\_Sanden40, [7](#)

MODELS\_Sanden80, [7](#)

MODELS\_Stiebel220E, [7](#)

MODELS\_basicIntegrated, [7](#)

MODELS\_externalTest, [7](#)

MODELS\_restankHugeUA, [7](#)

MODELS\_restankNoUA, [7](#)

MODELS\_restankRealistic, [7](#)

TYPE\_compressor, [7](#)

TYPE\_none, [7](#)

TYPE\_resistance, [7](#)

UNITS\_BTU, [7](#)

UNITS\_BTUperHrF, [7](#)

UNITS\_C, [7](#)

UNITS\_F, [7](#)

UNITS\_GAL, [7](#)

UNITS\_KJ, [7](#)

UNITS\_KWH, [7](#)

UNITS\_L, [7](#)

UNITS\_kJperHrC, [7](#)

VRB\_emetic, [8](#)

VRB\_minuteOut, [8](#)

VRB\_reluctant, [8](#)

VRB\_silent, [8](#)

VRB\_typical, [8](#)

HEATDIST\_MINVALUE  
    HPWH, [12](#)

HEATSOURCE\_TYPE  
    HPWH, [6](#)

HPWH, [4](#)

    ~HPWH, [8](#)

CONDENSITY\_SIZE, [12](#)

DRMODES, 6  
 getEnergyRemovedFromEnvironment, 8  
 getNthHeatSourceEnergyInput, 8  
 getNthHeatSourceEnergyOutput, 8, 9  
 getNthHeatSourceRunTime, 9  
 getNthHeatSourceType, 9  
 getNthSimTcouple, 9  
 getNumHeatSources, 9  
 getNumNodes, 9  
 getOutletTemp, 9  
 getSetpoint, 9  
 getStandbyLosses, 9  
 getTankNodeTemp, 10  
 getVersion, 10  
 HEATDIST\_MINVALUE, 12  
 HEATSOURCE\_TYPE, 6  
 HPWH, 8  
 HPWH\_ABORT, 12  
 HPWHinit\_file, 10  
 HPWHinit\_presets, 10  
 HPWHinit\_resTank, 10  
 HPWH, 8  
 isNthHeatSourceRunning, 10  
 isSetpointFixed, 11  
 MAXOUTSTRING, 13  
 MODELS, 7  
 operator=, 11  
 printHeatSourceInfo, 11  
 printTankTemps, 11  
 resetTankToSetpoint, 11  
 runNSteps, 11  
 runOneStep, 11  
 setAirFlowFreedom, 11  
 setDoTempDepression, 11  
 setInletT, 12  
 setMessageCallback, 12  
 setSetpoint, 12  
 setTankSize, 12  
 setUA, 12  
 setVerbosity, 12  
 UNITS, 7  
 VERBOSITY, 7  
 WriteCSVRow, 12  
 HPWH::HeatSource, 3  
   addHeat, 3  
   disengageHeatSource, 3  
   engageHeatSource, 4  
   HeatSource, 3  
   isEngaged, 4  
   setCondensity, 4  
   setupAsResistiveElement, 4  
   shouldHeat, 4  
   shutsOff, 4  
 HPWH\_ABORT  
   HPWH, 12  
 HPWHinit\_file  
   HPWH, 10  
 HPWHinit\_presets  
   HPWH, 10  
   HPWHinit\_resTank  
     HPWH, 10  
 HeatSource  
   HPWH::HeatSource, 3  
 isEngaged  
   HPWH::HeatSource, 4  
 isNthHeatSourceRunning  
   HPWH, 10  
 isSetpointFixed  
   HPWH, 11  
 MODELS\_AOSmithHPTU50  
   HPWH, 7  
 MODELS\_AOSmithHPTU66  
   HPWH, 7  
 MODELS\_AOSmithHPTU80  
   HPWH, 7  
 MODELS\_AOSmithPHPT60  
   HPWH, 7  
 MODELS\_AOSmithPHPT80  
   HPWH, 7  
 MODELS\_CustomFile  
   HPWH, 7  
 MODELS\_CustomResTank  
   HPWH, 7  
 MODELS\_GE2012  
   HPWH, 7  
 MODELS\_GE2014  
   HPWH, 7  
 MODELS\_GE2014STDMode  
   HPWH, 7  
 MODELS\_Generic1  
   HPWH, 7  
 MODELS\_Generic2  
   HPWH, 7  
 MODELS\_Generic3  
   HPWH, 7  
 MODELS\_RheemHB50  
   HPWH, 7  
 MODELS\_Sanden40  
   HPWH, 7  
 MODELS\_Sanden80  
   HPWH, 7  
 MODELS\_Stiebel220E  
   HPWH, 7  
 MODELS\_basicIntegrated  
   HPWH, 7  
 MODELS\_externalTest  
   HPWH, 7  
 MODELS\_restankHugeUA  
   HPWH, 7  
 MODELS\_restankNoUA  
   HPWH, 7  
 MODELS\_restankRealistic  
   HPWH, 7  
 MAXOUTSTRING  
   HPWH, 13

MODELS  
     HPWH, 7  
 operator=  
     HPWH, 11  
 printHeatSourceInfo  
     HPWH, 11  
 printTankTemps  
     HPWH, 11  
 resetTankToSetpoint  
     HPWH, 11  
 runNSteps  
     HPWH, 11  
 runOneStep  
     HPWH, 11  
 setAirFlowFreedom  
     HPWH, 11  
 setCondensity  
     HPWH::HeatSource, 4  
 setDoTempDepression  
     HPWH, 11  
 setInletT  
     HPWH, 12  
 setMessageCallback  
     HPWH, 12  
 setSetpoint  
     HPWH, 12  
 setTankSize  
     HPWH, 12  
 setUA  
     HPWH, 12  
 setVerbosity  
     HPWH, 12  
 setupAsResistiveElement  
     HPWH::HeatSource, 4  
 shouldHeat  
     HPWH::HeatSource, 4  
 shutsOff  
     HPWH::HeatSource, 4  
 TYPE\_compressor  
     HPWH, 7  
 TYPE\_none  
     HPWH, 7  
 TYPE\_resistance  
     HPWH, 7  
 UNITS\_BTU  
     HPWH, 7  
 UNITS\_BTUperHrF  
     HPWH, 7  
 UNITS\_C  
     HPWH, 7  
 UNITS\_F  
     HPWH, 7  
 UNITS\_GAL  
     HPWH, 7  
 UNITS\_KJ  
     HPWH, 7  
 UNITS\_KWH  
     HPWH, 7  
 UNITS\_L  
     HPWH, 7  
 UNITS\_kJperHrC  
     HPWH, 7  
 UNITS  
     HPWH, 7  
 VRB\_emetic  
     HPWH, 8  
 VRB\_minuteOut  
     HPWH, 8  
 VRB\_reluctant  
     HPWH, 8  
 VRB\_silent  
     HPWH, 8  
 VRB\_typical  
     HPWH, 8  
 VERBOSITY  
     HPWH, 7  
 WriteCSVRow  
     HPWH, 12