

HPWHsim

1.3.0

Generated by Doxygen 1.8.6

Fri Jul 29 2016 12:07:28

Contents

1	Class Index	1
1.1	Class List	1
2	Class Documentation	3
2.1	HPWH::HeatSource Class Reference	3
2.1.1	Constructor & Destructor Documentation	3
2.1.1.1	HeatSource	3
2.1.1.2	HeatSource	3
2.1.2	Member Function Documentation	4
2.1.2.1	addHeat	4
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	7
2.2.2.1	DRMODES	7
2.2.2.2	HEATSOURCE_TYPE	7
2.2.2.3	MODELS	7
2.2.2.4	UNITS	8
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	9
2.2.4.1	getEnergyRemovedFromEnvironment	9

2.2.4.2	getNthHeatSourceEnergyInput	9
2.2.4.3	getNthHeatSourceEnergyInput	9
2.2.4.4	getNthHeatSourceEnergyOutput	9
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	10
2.2.4.11	getNumNodes	10
2.2.4.12	getOutletTemp	10
2.2.4.13	getSetpoint	10
2.2.4.14	getStandbyLosses	10
2.2.4.15	getTankHeatContent_kJ	10
2.2.4.16	getTankNodeTemp	10
2.2.4.17	getTankNodeTemp	10
2.2.4.18	getVersion	10
2.2.4.19	HPWHinit_file	10
2.2.4.20	HPWHinit_genericHPWH	11
2.2.4.21	HPWHinit_presets	11
2.2.4.22	HPWHinit_resTank	11
2.2.4.23	HPWHinit_resTank	11
2.2.4.24	isNthHeatSourceRunning	11
2.2.4.25	isSetpointFixed	11
2.2.4.26	operator=	11
2.2.4.27	printHeatSourceInfo	11
2.2.4.28	printTankTemps	11
2.2.4.29	resetTankToSetpoint	12
2.2.4.30	runNSteps	12
2.2.4.31	runOneStep	12
2.2.4.32	runOneStep	12
2.2.4.33	setAirFlowFreedom	12
2.2.4.34	setDoTempDepression	12
2.2.4.35	setInletT	12
2.2.4.36	setMessageCallback	12
2.2.4.37	setSetpoint	12
2.2.4.38	setSetpoint	13
2.2.4.39	setTankSize	13
2.2.4.40	setUA	13
2.2.4.41	setVerbosity	13

2.2.4.42	WriteCSVRow	13
2.2.5	Member Data Documentation	13
2.2.5.1	CONDENSITY_SIZE	13
2.2.5.2	HEATDIST_MINVALUE	13
2.2.5.3	HPWH_ABORT	13
2.2.5.4	MAXOUTSTRING	13
2.2.5.5	UNINITIALIZED_LOCATIONTEMP	13
 Index		 14

Chapter 1

Class Index

1.1 Class List

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

HPWH::HeatSource	3
HPWH	4

Chapter 2

Class Documentation

2.1 HPWH::HeatSource Class Reference

Public Member Functions

- [HeatSource](#) ()
- [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 () [inline]

default constructor, does not create a useful [HeatSource](#)

2.1.1.2 HPWH::HeatSource::HeatSource ([HPWH](#) * *parentHPWH*)

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_GE2014STDMode_80` = 113, `MODELS_GE2014` = 112, `MODELS_GE2014_80` = 114, `MODELS_Sanden40` = 120,
`MODELS_Sanden80` = 121, `MODELS_RheemHB50` = 140, `MODELS_Stiebel220E` = 150, `MODELS_Generic1` = 160,
`MODELS_Generic2` = 161, `MODELS_Generic3` = 162, `MODELS_UEF2generic` = 170, `MODELS_genericCustomUEF` = 171,
`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 `HPWHinit_genericHPWH` (double tankVol_L, double energyFactor, double resUse_C)
- 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
- double [getTankHeatContent_kJ](#) () 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** = 3
- static const std::string **version_maint** = "0"
- 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_GE2014STDMode_80 2014 GE model run in standard mode, 80 gallon unit

MODELS_GE2014 2014 GE model run in the efficiency mode

MODELS_GE2014_80 2014 GE model run in the efficiency mode, 80 gallon unit

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_UEF2generic UEF 2.0, modified GE2014STDMode case

MODELS_genericCustomUEF used for creating "generic" model with custom uef

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::getTankHeatContent_kJ () const

get the heat content of the tank, relative to zero celsius returns using kilojoules

2.2.4.16 double HPWH::getTankNodeTemp (int nodeNum) const**Parameters**

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

2.2.4.17 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

2.2.4.18 string HPWH::getVersion () [static]

This function returns a string with the current version number

2.2.4.19 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

2.2.4.20 `int HPWH::HPWHinit_genericHPWH (double tankVol_L, double energyFactor, double resUse_C)`

This function will initialize a [HPWH](#) object to be a non-specific [HPWH](#) model with an energy factor as specified. Since energy factor is not strongly correlated with energy use, most settings are taken from the GE2015_STDMode model.

2.2.4.21 `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

2.2.4.22 `int HPWH::HPWHinit_resTank ()`

Default resistance tank, EF 0.95, volume 47.5

2.2.4.23 `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.

2.2.4.24 `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

2.2.4.25 `bool HPWH::isSetpointFixed ()`

is the setpoint allowed to be changed

2.2.4.26 `HPWH & HPWH::operator= (const HPWH & hpwh)`

assignment operator

2.2.4.27 `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

2.2.4.28 `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

2.2.4.29 int HPWH::resetTankToSetpoint ()

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

2.2.4.30 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

2.2.4.31 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

2.2.4.32 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

2.2.4.33 int HPWH::setAirFlowFreedom (double *fanFraction*)

This is a simple setter for the AirFlowFreedom

2.2.4.34 int HPWH::setDoTempDepression (bool *doTempDepress*)

This is a simple setter for the temperature depression option

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

Setters for the what are typically input variables

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

sets the function to be used for message passing

2.2.4.37 int HPWH::setSetpoint (double *newSetpoint*)

Parameters

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

2.2.4.38 `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

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

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

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

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

2.2.4.41 `void HPWH::setVerbosity (VERBOSITY hpwhVrb)`

sets the verbosity to the specified level

2.2.4.42 `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, [4](#)

CONDENSITY_SIZE

HPWH, [13](#)

DR_ALLOW

HPWH, [7](#)

DR_BLOCK

HPWH, [7](#)

DR_ENGAGE

HPWH, [7](#)

DRMODES

HPWH, [7](#)

disengageHeatSource

HPWH::HeatSource, [4](#)

engageHeatSource

HPWH::HeatSource, [4](#)

getEnergyRemovedFromEnvironment

HPWH, [9](#)

getNthHeatSourceEnergyInput

HPWH, [9](#)

getNthHeatSourceEnergyOutput

HPWH, [9](#)

getNthHeatSourceRunTime

HPWH, [9](#)

getNthHeatSourceType

HPWH, [9](#)

getNthSimTcouple

HPWH, [9](#)

getNumHeatSources

HPWH, [9](#)

getNumNodes

HPWH, [10](#)

getOutletTemp

HPWH, [10](#)

getSetpoint

HPWH, [10](#)

getStandbyLosses

HPWH, [10](#)

getTankHeatContent_kJ

HPWH, [10](#)

getTankNodeTemp

HPWH, [10](#)

getVersion

HPWH, [10](#)

HPWH

DR_ALLOW, [7](#)

DR_BLOCK, [7](#)

DR_ENGAGE, [7](#)

MODELS_AOSmithHPTU50, [7](#)

MODELS_AOSmithHPTU66, [7](#)

MODELS_AOSmithHPTU80, [7](#)

MODELS_AOSmithPHPT60, [7](#)

MODELS_AOSmithPHPT80, [7](#)

MODELS_CustomFile, [8](#)

MODELS_CustomResTank, [8](#)

MODELS_GE2012, [7](#)

MODELS_GE2014, [7](#)

MODELS_GE2014_80, [7](#)

MODELS_GE2014STDMode, [7](#)

MODELS_GE2014STDMode_80, [7](#)

MODELS_Generic1, [7](#)

MODELS_Generic2, [7](#)

MODELS_Generic3, [7](#)

MODELS_RheemHB50, [7](#)

MODELS_Sanden40, [7](#)

MODELS_Sanden80, [7](#)

MODELS_Stiebel220E, [7](#)

MODELS_UEF2generic, [7](#)

MODELS_basicIntegrated, [7](#)

MODELS_externalTest, [7](#)

MODELS_genericCustomUEF, [8](#)

MODELS_restankHugeUA, [7](#)

MODELS_restankNoUA, [7](#)

MODELS_restankRealistic, [7](#)

TYPE_compressor, [7](#)

TYPE_none, [7](#)

TYPE_resistance, [7](#)

UNITS_BTU, [8](#)

UNITS_BTUperHrF, [8](#)

UNITS_C, [8](#)

UNITS_F, [8](#)

UNITS_GAL, [8](#)

UNITS_KJ, [8](#)

UNITS_KWH, [8](#)

UNITS_L, [8](#)

UNITS_kJperHrC, [8](#)

VRB_emetic, [8](#)

VRB_minuteOut, [8](#)

VRB_reluctant, [8](#)

VRB_silent, [8](#)

VRB_typical, [8](#)

HEATDIST_MINVALUE

HPWH, [13](#)

HEATSOURCE_TYPE
 HPWH, 7
 HPWH, 4
 ~HPWH, 8
 CONDENSITY_SIZE, 13
 DRMODES, 7
 getEnergyRemovedFromEnvironment, 9
 getNthHeatSourceEnergyInput, 9
 getNthHeatSourceEnergyOutput, 9
 getNthHeatSourceRunTime, 9
 getNthHeatSourceType, 9
 getNthSimTcouple, 9
 getNumHeatSources, 9
 getNumNodes, 10
 getOutletTemp, 10
 getSetpoint, 10
 getStandbyLosses, 10
 getTankHeatContent_kJ, 10
 getTankNodeTemp, 10
 getVersion, 10
 HEATDIST_MINVALUE, 13
 HEATSOURCE_TYPE, 7
 HPWH, 8
 HPWH_ABORT, 13
 HPWHinit_file, 10
 HPWHinit_genericHPWH, 10
 HPWHinit_presets, 11
 HPWHinit_resTank, 11
 HPWH, 8
 isNthHeatSourceRunning, 11
 isSetpointFixed, 11
 MAXOUTSTRING, 13
 MODELS, 7
 operator=, 11
 printHeatSourceInfo, 11
 printTankTemps, 11
 resetTankToSetpoint, 11
 runNSteps, 12
 runOneStep, 12
 setAirFlowFreedom, 12
 setDoTempDepression, 12
 setInletT, 12
 setMessageCallback, 12
 setSetpoint, 12
 setTankSize, 13
 setUA, 13
 setVerbosity, 13
 UNITS, 8
 VERBOSEITY, 8
 WriteCSVRow, 13
 HPWH::HeatSource, 3
 addHeat, 4
 disengageHeatSource, 4
 engageHeatSource, 4
 HeatSource, 3
 isEngaged, 4
 setCondensity, 4
 setupAsResistiveElement, 4
 shouldHeat, 4
 shutsOff, 4
 HPWH_ABORT
 HPWH, 13
 HPWHinit_file
 HPWH, 10
 HPWHinit_genericHPWH
 HPWH, 10
 HPWHinit_presets
 HPWH, 11
 HPWHinit_resTank
 HPWH, 11
 HeatSource
 HPWH::HeatSource, 3

 isEngaged
 HPWH::HeatSource, 4
 isNthHeatSourceRunning
 HPWH, 11
 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, 8
 MODELS_CustomResTank
 HPWH, 8
 MODELS_GE2012
 HPWH, 7
 MODELS_GE2014
 HPWH, 7
 MODELS_GE2014_80
 HPWH, 7
 MODELS_GE2014STDMode
 HPWH, 7
 MODELS_GE2014STDMode_80
 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_UEF2generic
 HPWH, [7](#)
 MODELS_basicIntegrated
 HPWH, [7](#)
 MODELS_externalTest
 HPWH, [7](#)
 MODELS_genericCustomUEF
 HPWH, [8](#)
 MODELS_restantkHugeUA
 HPWH, [7](#)
 MODELS_restantkNoUA
 HPWH, [7](#)
 MODELS_restantkRealistic
 HPWH, [7](#)
 MAXOUTSTRING
 HPWH, [13](#)
 MODELS
 HPWH, [7](#)

 operator=
 HPWH, [11](#)

 printHeatSourceInfo
 HPWH, [11](#)
 printTankTemps
 HPWH, [11](#)

 resetTankToSetpoint
 HPWH, [11](#)
 runNSteps
 HPWH, [12](#)
 runOneStep
 HPWH, [12](#)

 setAirFlowFreedom
 HPWH, [12](#)
 setCondensity
 HPWH::HeatSource, [4](#)
 setDoTempDepression
 HPWH, [12](#)
 setInletT
 HPWH, [12](#)
 setMessageCallback
 HPWH, [12](#)
 setSetpoint
 HPWH, [12](#)
 setTankSize
 HPWH, [13](#)
 setUA
 HPWH, [13](#)
 setVerbosity
 HPWH, [13](#)
 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, [8](#)
 UNITS_BTUperHrF
 HPWH, [8](#)
 UNITS_C
 HPWH, [8](#)
 UNITS_F
 HPWH, [8](#)
 UNITS_GAL
 HPWH, [8](#)
 UNITS_KJ
 HPWH, [8](#)
 UNITS_KWH
 HPWH, [8](#)
 UNITS_L
 HPWH, [8](#)
 UNITS_kJperHrC
 HPWH, [8](#)
 UNITS
 HPWH, [8](#)

 VRB_emetic
 HPWH, [8](#)
 VRB_minuteOut
 HPWH, [8](#)
 VRB_reluctant
 HPWH, [8](#)
 VRB_silent
 HPWH, [8](#)
 VRB_typical
 HPWH, [8](#)
 VERBOSITY
 HPWH, [8](#)

 WriteCSVRow
 HPWH, [13](#)