Spec:Residential

From GridLAB-D Wiki

TODO: This document needs to be pruned down the specifications only. --Dchassin 20:34, 24 November 2011 (UTC)

Residential Overview

A general purpose enduse object is provided that incorporate a simple translation of a schedule to a loadshape to an enduse load. The enduse is linked to a circuit (which may be either line-to-line or line-to-neutral).

The residential_enduse class is defined as

```
class residential_enduse {
	loadshape shape;
	complex demand[kVA]; // the peak pow
	complex energy[kVAh]; // the total e
	complex total_power[kVA]; // the tot
	double heatgain[Btu/h]; // the heat 
	double heatgain_fraction; // the fra
	double current_fraction; // the frac
	double impedance_fraction; // the fr
	double power_fraction; // the fracti
	double power_factor; // the power fa
	complex constant_power[kVA]; // the 
	complex constant_current[kVA]; // th
	complex constant_admittance[kVA]; //
	double voltage_factor[pu]; // the vo
	set {IS220=1} configuration; // the 
	enumeration {OFF=-1, NORMAL=0, ON=1}
	enumeration {ON=1, OFF=0, UNKNOWN=-1
	complex total[kVA]; // (DEPRECATED) 
	complex power[kVA]; // (DEPRECATED) 
	complex current[kVA]; // (DEPRECATED
	complex admittance[kVA]; // (DEPRECA
}
```

The various end use appliances within the residential module have a common `enduse` member. The four component values of this structure are published consistently by the house as the enduse load name (e.g., *lights*, *plugs*) The individual properties should be used for internal reference for a given appliance and aggregated to the "enduse_load" property, which may be

Contents

used for load calculations by the house or by other objects.

The central importance of the enduse structure is that these four principle properties (power, demand, energy, and heatgain) must be updated by the object using a call to *gl_sync_enduse* so that when the enduse is attached to a house circuit panel the load accumulate correctly. If the properties are not published and updated, the house will halt the simulator.

The override property is implemented on a case-by-case between objects, and is meant to provide a mechanism for other objects to force a residential enduse to immediately activate or deactivate. The house, for example, will ignore its previous state and either immediately start heating or cooling, or immediately stop heating or cooling, based on if the house could be in such a state. For example, too-cold houses will not stop heating or start cooling, no matter the signal.

The power_state property is meant to indicate to other devices whether the enduse is currently drawing power or not. This is primarily used by the market module.

Implicit enduses

A number of enduses can be implicitly defined by listing them in the residential module's global parameter *implicit_enduses*. If the parameter is not specified in the module directive, all implicit enduses are activated as shown in Table 1. Implicit enduses that may be specified are:

```
module residential {
  implicit_enduses LIGHTS|CLOTHESWASHER|WATERHEATER|REFRIGERATOR|DRYER|FREEZER|DISHWASHER;
}
```

but others are expected any time, so please consult the *--modhelp residential* output for what is currently supported.

Table 1 - Implicit enduses

End use	Type	Schedule	Parameters
Lights	analog	residential-lights-default	power: 760 W
Plugs	analog	residential-plugs-default	power: 360 W
Clotheswasher	pulsed	residential-clotheswasher-default	energy: 750 Wh count: 0.25 power: 1 kW stdev: 150 W
Waterheater	frequency modulated	residential-waterheater-default	energy: 1 kWh count: 1 power: 5 kW stdev: 500 W
Refrigerator	frequency modulated	residential-refrigerator-default	energy: 1 kWh count: 25 power: 750 W stdev: 100 W
Dryer	pulsed	residential-dryer-default	energy: 2.5 kWh count: 0.25 power: 5 kW tdev: 0.5 kW
Freezer	frequency modulated	residential-freezer-default	energy: 750 Wh count: 25 power: 500 W stdev: 50 W
Dishwasher	pulsed	residential-dishwasher-default	energy: 1.0 kWh power: 1.0 kW count: 1.0 stdev: 150 W
Range	pulsed	residential-range-default	energy: 1.0 kWh power: 500 W count: 1.0 stdev: 95 W
Microwave	pulsed	residential-microwave-default	energy: 1.0 kWh power: 200 W count: 1.0 stdev: 40 W

Using the House

The house object is a dual purpose class. It first operates to aggregate and contain the effects of the various appliance loads, but also contains an appliance, the HVAC system, which it uses to control the thermal effects of the solar input and the electrical load heat of the building.

Default House

PLEASE NOTE: This section on the default house parameters will be deprecated or updated in the near future. Please go to http://sourceforge.net/apps/mediawiki/gridlab-d/index.php?title=Residential_module_user%27s_guide for a further, more complete description of the house model. All other information on this page should be up-to-date.

House Properties

PLEASE NOTE: This section on the default house parameters will be deprecated or updated in the near future. Please go to http://sourceforge.net/apps/mediawiki/gridlab-d/index.php?title=Residential_module_user%27s_guide for a further, more complete description of the house model. All other information on this page should be up-to-date.

House State of Development

House is considered a stable model, with many features.

Using the dishwasher model

Description here.

Default dishwasher

Default here.

Dishwasher Properties

Properties here.

Dishwasher State of Development

Dishwasher is considered an experimental model, and may not function correctly at this time.

Using the evcharger model

Description here.

Default evcharger

Default here.

evcharger Properties

Properties here.

evcharger State of Development

evcharger is considered an experimental model, and while functionality exists, it is very limited.

Using the Freezer

The freezer is modeled by determining the thermal capacity of the hypothetical contents of the freezer and estimating the thermal gains of the freezer cavity through its insulation. Heat gain through intermittent door opening is not factored in at this point.

From a simulation view, the freezer gradually absorbs heat from the house, and will activate its cooling system motor when the temperature exceeds a certain point.

Default Freezer

A default freezer will be defined with

```
object freezer {
}
```

will be initialized with the values of

```
object freezer {
   size random.uniform(20,40);
   thermostat_deadband random.uniform(2,3);
   setpoint random.uniform(10,20);
   UA 6.5;
   power_factor 0.95;
   rated_capacity (size * 34.0);
}
```

Freezer Properties

Property Name	Type	Unit	Default Value	Description
size	double	cu ft	20 - 40 cu ft	Storage volume of the freezer
rated_capacity	double	BTU / hr	10 BTU/h per cu ft	Cooling capacity of the freezer under optimum conditions.
power_factor	double	ratio	0.95	...
temperature	double	degF	10.0 - setpoint	Read-only. Air temperature inside the freezer.

setpoint	double	degF	10.0 - 20.0	The temperature the thermostat is set at to stay colder than
deadband	double	degF	2.0 - 3.0	The 'slack' in the thermostat, and the temperature to cool the freezer by when the thermostat starts the cooling cycle.
next_time	timestamp	sec	-	The next time that the internal state of the freezer will change due to thermal conditions.
output	double	???	-	Read-only. Heat rate from the cooling system.
UA	double	BTU*hr/degF	6.5	The relative heat loss of the freezer across the surface of its housing. Smaller values indicate better insulation.
state	enumeration	OFF, ON	OFF	Read-only. Current state of the freezer cooling motor.
enduse_load	complex	kW	-	Read-only. Current power consumption by the freezer.
constant_power	complex	kW	-	Read-only. Constant power part of the current power draw.
constant_current	complex	A	-	Read-only. Constant current part of the current power draw.
constant_admittance	complex	1/Ohm	-	Read-only. Constant resistance part of the current power draw.
internal_gains	double	kW	-	Read-only. The heat created and released into the air by this appliance.
energy_meter	double	kWh	-	The energy consumed during the running life of the appliance.

Freezer State of Development

Freezer is considered an experimental model, and may not function correctly at this time.

Using the lights model

Description here.

Default lights

Default here.

lights Properties

Properties here.

Lights State of Development

Lights is considered a simple, stable model.

Using the microwave model

Description here.

Default microwave

Default here.

microwave Properties

Properties here.

Microwave State of Development

Microwave is considered an experimental model, and may not function correctly at this time.

Using the occupantload model

Description here.

Default occupantload

Default here.

occupantload Properties

Properties here.

occupantload State of Development

occupantload is considered an experimental model, and while most of the functionality should exist, it has not been thoroughly tested.

Using the plugload model

Description here.

Default plugload

Default here.

plugload Properties

Properties here.

Plugload State of Development

Plugload is considered a simple, stable model.

Using the range model

Description here.

Default range

Default here.

range Properties

Properties here.

Range State of Development

Range is considered an experimental model, and may not function correctly at this time.

Using the Refrigerator

The Refrigerator copies the behaviors and the properties from the Freezer

Default Refrigerator

Refrigerator Properties

See Freezer Properties

Refrigerator State of Development

Refrigerator is considered an experimental model, and may not function correctly at this time.

Using the thermal_storage model

The thermal storage model is based on the specifications of Ice Energy's Ice Bear system. It is a 5 ton equivalent unit that is to be used in conjunction with a normal HVAC unit. The Ice Bear unit is used for peak load shifting and does so by storing thermal energy in the form of ice. The ice is made at night and then used during the day to reduce or eliminate the use of the HVAC compressor, thereby reducing the HVAC load to about 10% of normal during peak hours.

Default thermal_storage

An empty thermal storage object, along the lines of

```
object thermal_storage {}
```

will be constructed into a semi-consistent state. Assuming a 5 ton (60,000 Btu/hr) unit the "default thermal storage" ends up being similar to

```
schedule recharge_sched {
   * 0-10 * * * 1.0;
   * 11-20 * * * 0.0;
   * 21-23 * * * 1.0;
}

schedule discharge_sched {
   * 0-10 * * * 0.0;
   * 11-20 * * * 1.0;
   * 21-23 * * * 0.0;
}

object thermal_storage {
   total_capacity 360000;
   stored_capacity 360000;
   recharge_power 3.360;
   discharge_power 0.300;
   recharge_pf 0.97;
   discharge_pf 1;
   recharge_time recharge_sched*1;
   discharge_time discharge_sched*1;
   discharge_rate 60000;
   k 0;
}
```

Note that setting both "stored_capacity" and "SOC" will raise a warning, since both values are attempted to define the initial capacity of the thermal_storage object. If SOC is used, the stored_capacity will be set to (SOC / 100 * total_capacity).

Any properties that are not set explicitly will carry these default values, with the exception of the total_capacity, discharge_power, recharge_power and discharge_rate, which will have values based on the designed_cooling_capacity of house_e and scaled appropriately based on the values listed here.

thermal_storage Properties

Table 1 - Thermal Storage Properties

Property Name	Type	Unit	Description
total_capacity	double	Btu	The total capacity of energy storage of the unit. When left to default, it is scaled based on the HVAC sizing in house_e.
stored_capacity	double	Btu	The amount of energy stored in the unit at the start of the simulation. If this exceeds the total_capacity, it will be set equal to the total_capacity. If SOC (state of charge) is also set, SOC is the dominant value and will be used instead.
recharge_power	double	kW	The rated power required to run the compressor and charge the unit (make ice).
discharge_power	double	kW	The rated power required to run the pump and discharge the unit (melt the ice).
recharge_pf	double	NA	The rated power factor of the compressor to charge the unit (make ice).
discharge_pf	double	NA	The rated power factor of the pump to discharge the unit (melt the ice).
discharge_schedule_type	enum	NA	Specifies the use of either the "INTERNAL" or "EXTERNAL" schedule for the discharge (INTERNAL = default, EXTERNAL = user defined)
recharge_schedule_type	enum	NA	Specifies the use of either the "INTERNAL" or "EXTERNAL" schedule for the recharge (INTERNAL = default, EXTERNAL = user defined)
recharge_time	double	NA	The time schedule indicating the hours of operation for the recharge cycle (0 = OFF, 1 = ON). The recharge and discharge cycles can not overlap. The model will default to a recharge cycle in the event that both are set to be on.
discharge_time	double	NA	The time schedule indicating the hours of operation for the discharge cycle (0 = OFF, 1 = ON). The recharge and discharge cycles can not overlap. The model will default to a recharge cycle in the event that both are set to be on.
discharge_rate	double	Btu/hr	The rated capacity of the unit as it would relate to a normal HVAC unit (i.e. a 5 ton unit to cool a house). When left to default, it is scaled based on the HVAC sizing in house_e.
SOC	double	%	The state of charge of the system in percent of ice energy available. If store capacity is also set, SOC is the dominant value and will be used.
k	double	W/m/°C	The coefficient of thermal conductivity in Watts per meter per °C.

Thermal Storage State of Development

thermal_storage is considered a stable model and contains many features. It is relatively new and subject to change as new features may be added at a later date.

Using the washer model

Description here.

Default washer

Default here.

washer Properties

Properties here.

Washer State of Development

Washer is considered an experimental model, and may not function correctly at this time.

Using the Waterheater

The waterheater is modeled as either a one-node or a two-node body of heat with thermal resistance between the interior and the exterior of the model.

Default Waterheater

An empty waterheater object, along the lines of

```
object waterheater { }
```

will be constructed into a semi-consistant state. The "default waterheater" ends up being similar to

```
object waterheater {
   tank_volume 50.0 gal;
   tank_diameter 1.5 ft;
   inlet_water_temperature 60.0 degF;
   location GARAGE;
   heat_mode ELECTRIC;
   tank_setpoint random.normal(130,10); // bound [100, 160]
   thermostat_deadband (1 + random.normal(2,1)); // bound [1, 10]
   tank_UA random.normal(2.0, 0.2); // bound (1, inf)
   heating_element_capacity 4500 W;
}
```

Any properties that are not set explicitly will carry these default values.

Waterheater Properties

Property Name	Type	Unit	Description
tank_volume	double	gallons	The water volume of the water tank.
tank_UA	double	BTU/hour	The product of the U-value of the tank's insulation and the surface area of the tank, assuming R values of about 13.
tank_diameter	double	feet	The diameter of the water tank, influences heat loss calculations.
water_demand	double	gallons/minute	Hot water consumption. Constant unless controlled by a Player object.
heating_element_capacity	double	Watts	The rate at which the waterheater heating element will dump thermal energy into the water tank.
inlet_water_temperature	double	degF	The temperature of the cold water entering the bottom of the waterheater to replace any hot water drawn out the top of the tank.
heat_mode	enumeration		"ELECTRIC" or "GASHEAT". Determines the method that heat is added to the water tank.
location	enumeration		"INSIDE" or "GARAGE". Placement determines if thermal losses from the water heater wind up heating up the house, and if the outside temperature influences the effective temperature for heat loss.
tank_setpoint	double	degF	The target temperature at which the heating elements will click on and off in the waterheater.
thermostat_deadband	double	degF	The number of degrees to heat the water when needed. Influences when the water heating element will turn on and turn off.
meter	double	kilowatt-hours	The total power consumed by the water heater during the simulation.
temperature	double	degF	The temperature of the hot water in the tank.
height	double	feet	The height of the hot water tank.
enduse_load	complex	kilowatts	The current power draw of the water heater. Required by the house to attach the water heater to the circuit panel.

constant_power	complex	kilowatts	The constant power draw of the water heater. No effect ~ modify the heating_element_capacity.
constant_current	complex	amps	The constant current draw of the water heater. No effect.
constant_admittance	complex	1/Ohm	The constant admittance of power across the water heater. No effect.
internal_gains	double	kilowatts	The heat loss for the current timestep from the water heater to the water tank's location.
gas_fan_power	double	kW	The load of a running gas waterheater, primarily from any venting fan.
gas_standby_power	double	kW	The load of a gas waterheater in standby mode ~ digital logic attached to the thermostat, etc.

Waterheater State of Development

Waterheater is considered a stable model, with a fair amount of functionality.

Using the ZIPload model

Description here.

Default ZIPload

Default here.

ZIPload Properties

Properties here.

ZIPload State of Development

ZIPload is considered a simple, stable model, with many layers of functionality.

See also

- Residential module
    - User's Guide
    - Appliances
    - house class – Single-family home model.

    - residential_enduse class – Abstract residential end-use class.
    - occupantload – Residential occupants (sensible and latent heat).
    - ZIPload – Generic constant impedance/current/power end-use load.
- Technical Documents
    - Requirements
    - **Specifications**
    - Developer notes
    - Technical support document
    - Validation

- This page was last modified on 31 October 2012, at 19:42.