FLORIDA SOLAR ENERGY CENTER®

CONTRACT REPORT

Improving the Accuracy and Speed for Building America Benchmarking

FSEC-CR-1651-06

September 27, 2006

Submitted to:

U.S. Department of Energy
Cooperative Agreement No.
DE-FC26-99GO10478

Authors:

Robin Vieira
Lixing Gu
Raju Sen Sharma
Carlos Colon
Danny Parker

1679 Clearlake Road, Cocoa, FL 32922-5703 • Phone: 321-638-1000 • Fax: 321-638-1010
www.fsec.ucf.edu

A Research Institute of the University of Central Florida

DISCLAIMER

ACKNOWLEDGMENTS

This work is sponsored, in large part, by the US Department of Energy (DOE), Office of Energy Efficiency and Renewable Energy, Building Technologies Program under cooperative agreement number DE-FC26-99GO10478. This support does not constitute an endorsement by DOE of the views expressed in this report.

We appreciate the encouragement and support from Mr. George James, Mr. Ed Pollock and Mr. Bill Haslebacher of DOE.

Table of Contents

Abstract

FSEC has developed capabilities for modelers to more accurately and more readily perform Building America energy analysis through three efforts:

- Developing a software toolkit for creating the Building America Benchmark home for use by software programmers
- Developing a simple hot water distribution algorithm for potential inclusion in future Building America efforts
- Simulating different mechanical ventilation options for homes.

Executive Summary

The Building America Benchmarking process is time consuming. One must first enter the parameters of the prototype home design into the Building America Spreadsheet tool[1] to create the parameters of the Building America benchmark home and then use detailed software products to simulate both the benchmark home and the prototype home and then enter the results from these simulations into the spreadsheet to determine the resulting % improvement for the prototype.

To reduce this effort, FSEC has created a BA Toolkit that allows programmers to incorporate calls to functions and procedures that produce the Building America Benchmark characteristics. This will enable programmers to more easily incorporate Benchmark analysis into their software. The toolkit has been tested against the Department of Energy developed Building America Spreadsheet Tool and found to produce the same results.

The Benchmarking process goes into great detail to determine hot water use for the benchmark and the prototype homes. However, the benchmark process has not included hot water distribution effects, which can be larger than many of the water use differences currently painstakingly calculated. FSEC has developed a simple routine and verified it against measured data as a method to simulate distribution effects to a reasonable degree of accuracy. It shows that typical losses in a Miami home may represent an increase of 2.4 % in hot water energy use and also a slight increase in cooling energy. This routine can be incorporated into or run separately from other software should the Building America program decide to include this element.

FSEC has also accomplished another enhancement for Building America teams that facilitates extraction of ventilation fan energy use from DOE reports. As part of this effort, but outside of this funding, FSEC also added a capability within EnergyGauge USA to simulate mechanical air handler ventilation with a controller that closes a damper after a certain amount of runtime or turn on the blower to assure a minimum amount of runtime, or both. This report presents simulation results for controlling mechanical

1 *Building America Analysis Spreadsheet*, dated 05.03/06 found online at http://www.eere.energy.gov/buildings/building_america/pa_resources.html

ventilation via nine strategies. Fresh air provided by systems as well as energy use due to ventilation air flow and fan energy consumption can vary significantly depending oncontrol characteristics. Simple runtime vent systems may only bring in air 20% to 25% of the time on an annual average basis compared to continuous vent systems and if designed for small quantities of air will likely not provide much more outdoor air than simple infiltration in the wintertime when the natural driving forces are large. Ensuring that a runtime vent system operates 25% of every hour results in increased energy use due to increased fan use (4% and 13% of heating and cooling energy, respectively) in the modeled St. Louis example used in this study.

In summary, FSEC has added capabilities for modelers to more accurately and more readily perform energy analysis for Building America homes. These algorithms are listed in the report text and the appendix. Please contact the first author Rob Vieira (robin@fsec.ucf.edu) to obtain these algorithms in computer friendly format.

1.0 Introduction

The Building America program has developed their unique energy analysis procedure. This procedure is different from the HERS and Energy Star procedures and different from the IECC code. To help teams accomplish the energy analysis, the National Renewable Energy Laboratory (NREL) has developed a spreadsheet that allows users to enter prototype characteristics and obtain inputs to building simulation programs for software tools. This spreadsheet has been improved over time and is a powerful tool. However, there are still some potential problems with the process. Each user has to take the outputs and enter them into a building simulation tool. This is an additional process and requires instructions for each type of software tool used. Therefore, there is a reasonable chance for error or at least inconsistency even if the same spreadsheet and building simulation programs are implemented by two users. In an effort to improve consistency and accuracy, FSEC set out to develop a stand-alone benchmark tool that will allow programmers to obtain the benchmark home characteristics and incorporate them into their software. FSEC also set out to improve mechanical ventilation modeling options and hot water distribution analysis.

2.0 BA Benchmark ToolKit algorithms

After considering adding a number of features, and starting with large building decks and other options, the FSEC team decided to simply code the spreadsheet benchmark information with a few choices as to how to obtain the outputs. This method was chosen as it required the smallest number of inputs and as such may be more readily used. The program includes flexibility that will allow for calling each component routine so if someone is experimenting with new benchmarks or building codes, they could choose to use portions of the toolkit and program some new portions.

The BA benchmark algorithms are being delivered with a COM object[2] for easy access within most software tools. Four units were coded to produce the BA benchmark building/spreadsheet procedures – one for the envelope, one for the HVAC and water heating equipment, one for the appliances and one with some common math-type routines. This section of the report provides an overview of each program unit developed and assumes the reader has an understanding of the benchmark building procedure. Detailed input and output descriptions are located in Appendix A. For full understanding of the Building America Benchmark requirements the reader is encouraged to view the documents at:
http://www.eere.energy.gov/buildings/building_america/pa_resources.html

2.1 Envelope

2 **Component Object Model** (**COM**) is a Microsoft platform for software components introduced by Microsoft in 1993. It is used to enable interprocess communication and dynamic object creation in any programming language that supports the technology

Programmers may call a master envelope procedure that calls each envelope component routine and returns all of the benchmark home envelope characteristics for a given prototype home. This routine requires just ten inputs. It produces 29 outputs (See Appendix A for details on inputs and outputs), some include string fields which may not be needed for software programs. For example, based on the benchmark home output characteristics of SHGC and U-value, the BA-ToolKit will provide an estimate of the glass-type (Single, double, low-e double or low-e triple). For very simple programs, or complicated programs that model the sunlight through each glass layer this may be needed, as opposed to just the U-value and SHGC.

Alternatively to using the master envelope procedure, programmers can call each envelope component procedure or function. Thus, if they just wanted to know the window U-value for a home they could call that routine, which only requires the heating degree days to obtain the window U-value. Although it is envisioned that calling the master unit is easier, there may some programmers who prefer the component approach because of the way they present data or write building decks.

A helpful routine is included in the envelope section that is not called from the master as it would have required two additional inputs per assembly. The *GetInsRValue* routine calculates the cavity insulation R-value based on parallel heat transfer principles for an assembly. Given the overall U-value, the stud R-value and framing fraction and the R-value of the components in series with both the stud and the cavity insulation, the function returns the cavity insulation R-value. The cavity insulation is often used to describe an assembly, e.g, "..the house has R-11 walls, R-30 ceiling, etc." even though the total R-value of the assembly will be a different value. The BA benchmark routines as well as most performance based building codes simply provide an overall assembly U-value. This routine allows a programmer who knows the characteristics of the typical building assembly and has calculated the overall U-value using the BA-ToolKit to determine the cavity R value for the benchmark home. It should prove useful for building decks as well as any simple building programs.

2.2 Equipment

The equipment unit includes procedures for determining the benchmark home heating, cooling, air distribution and mechanical ventilation characteristics. The master routine requires 23 inputs. The master unit will call each of the component units once. As such, it is not recommended for multiple system homes. For those homes, calling each component separately is recommended. For example, the Benchmark heating characteristics will change if the prototype has an electric or gas system. The type of heating system is a required input. Interestingly, the BA benchmark cooling system procedure requires no inputs. The benchmark duct system requires the prototype home finished floor area, the number of stories, the predominant foundation type, and the number of returns.

The hot water calculation requires the monthly ambient temperatures and eight other inputs. Unlike other reference homes, the BA benchmark requires a number of

parameters to determine the hot water use per day. Other standards simply base it on the number of bedrooms, but the BA benchmark computes an average monthly value based on monthly average water mains, the hot water supply and delivery temperature, and expected dishwasher, clotheswasher, sink and shower/tub schedules based on the number of bedrooms as a surrogate for occupants.

The mechanical ventilation routines calculate the flow and power based on the number of bedrooms and finished floor area.

Detailed descriptions of inputs required for each function are included in Appendix A.

2.3 Appliances

The BA-Toolkit calculates Building America energy use parameters for the following appliances:

Clotheswashers
Dishwashers
Dryers
Hard-wired lighting
Plug-in Lighting
Range
Refrigeration
Miscellaneous (this represents non-lighting plug loads) and
Occupants.

Similar to the envelope and equipment units, the appliance unit consists of a master and individual calls. A home with a gas and electric dryer may need to call the individual routines, however, some programs may only be set up for one of each appliance input. Each appliance procedure returns a peak hour power use and a 24-hour fraction of peak schedule, annual energy use and total, latent and sensible fraction of energy released to the interior.

2.4 Miscellaneous algorithms

The BA-Misc unit contains simple math or comparison functions used by multiple other units.

2.5 The ToolKit Application for Testing

The Building America ToolKit code was tested against the latest NREL spreadsheet for all benchmark specifications. The test unit was written in Visual Basic Application as a part of the spreadsheet. The test program uses the Building America Prototype Input worksheet for inputs and the Calcs5 worksheet for weather data inputs. The BA-toolkit outputs are written to the Benchmark Outputs Test worksheet. The outputs are then

compared to the spreadsheet outputs to verify agreement. In order to aid a programmer, the output variable names are given to the right of the cell where the test software will produce the values.

3.0 A simplified hot water distribution system model

3.1 Introduction

Water heating in the U.S. is a major component of total energy consumption in buildings. In the residential sector water heating is about 11% of the total.[3] The Department of Energy (DOE) lists total primary energy consumption for residential water heating at 2.66 quads. Hot water use in residential buildings accounts for the second largest portion of residential energy consumption in the U.S., second to the energy used for space heating.

It has been estimated that, on average, hot water distribution losses can be in excess of 20% between storage and the end-use point.[4] As energy efficiency in buildings improves with technology advances and modern building practices, hot water heating energy can now reach as much as 32% of the energy used on a high performance home.[5] Although the efficiency of water heaters has been mandated by national standards, the efficiency of the distribution system has gone unaddressed. As such, it appears that there is much potential for energy savings in water heating systems by improving and optimizing the design of hot water distribution systems (HWDS).

Many complex factors contribute to heat losses in a hot water distribution system. In addition to the thermal conductivity of the pipe materials used in today's construction (i.e., copper, PEX and CPVC), the environment in which the pipe is routed plays an important role. In a recent study for the CEC, ORNL performed detailed simulations of typical HWDS installations and found significant line losses, especially in recirculating systems. [6]

Due to the complex heat losses of HWDS, models are needed to optimize HWDS by reducing heat losses. There are three models currently used to simulate thermal performance of hot water distribution systems: HWSim, ORNL-HWDS, and TRNSYS.

The HWSIM model,[7] originally developed in 1991 as part of Davis Energy Group's (DEG) original hot water research for the California Energy Commission, has been used since 1992 to develop hot water distribution loss assumptions in California's Residential Standards. The program has significant capabilities but also has shortcomings stemming

---

3 Guide for the Evaluation of Energy Savings Potential, Office of Building Technology, State and Community Programs (BTS), Department of Energy, Industry Interactive Procurement Sysem (IIPS), < http:/e-center.doe.gov>

4 California Energy Commission, Measure Analysis and Life-Cycle Cost (Part 1): 2005 California Building Energy Efficiency Standards, P400-02-011, April 2002

5 Building America Experts Meeting Highlights Opportunities for Hot Water energy Savings, July 2004 <http://www.eere.energy.gov/buildings/building_america/rh_0704_home_improve.html>

6 Wendt, R., Baskin, E. and D. Durfee, "Evaluation of Residential Hot Water Distribution Systems by Numeric Simulation", ORNL, May 2004

7 Note: We could not find documentations related to HWSim publicly. The model information is extracted from **Scope of Work: Water Heaters and Hot Water Distribution System,** April, 2005, led by Lawrence Berkeley National Lab

from the limited scope of the original development effort. In 2004, DEG obtained funding to enhance the program. Key improvements to the model include the ability to simulate distribution system performance under changing environmental conditions (can adjust inlet cold water temperature and pipe environment temperatures on a monthly basis), improved user interface, and enhanced heat loss algorithms.

ORNL has also developed a numerical model to estimate heat loss or gain from insulated and non-insulated hot water pipes.[8] The required inputs are pipe parameters, insulation properties, and water flow rates. It calculates energy use, water consumption, and waiting time at use points. The model has been used to evaluate impacts of alternative HWDS in prototypes of California houses. The model includes thermal mass impacts from water, piping and water flow rates. The model is limited to the study of hot water distribution systems but could be incorporated into a whole building models like DOE-2 and ***EnergyPlus***.

Using the Transient Energy System Simulation Tool, TRNSYS,[9] a simulation model was developed by NAHB to estimate energy consumption for hot water systems and to further simulate other system design options.[10] The simulation model was calibrated with heat-transfer coefficients determined by experimental results. The model requires water flow rates and assumes no thermal mass impacts. However, it is a whole building approach and is able to simulate interactions between a building and the HWDS. It was used to evaluate the use of demand water heating equipment in conjunction with various hot water piping configurations.

The first two models are used to study hot water distribution systems only and may not meet Building America program requirements for a whole building approach. The third model is a whole building approach; however, it does not include the important thermal mass impacts. The present effort is to develop a simplified HWDS model that includes dynamic impacts and which can be used in the DOE-2 program as an input function.

3.2 Simplified model development

The following simplifying assumptions are used in the model developed here:

- Water temperature is constant at a given cross section
- When a copper pipe is used, conductive resistance through the copper pipe wall is assumed to be negligible
- Water and copper pipe have the same temperature at a given cross section.
- Water and copper pipe temperature is a function of distance from the hot water source and the length of time the outlet (faucet or shower) is activated.

---

8 Wendt, R.; E. Baskin & D. Durfee, 2004, "Evaluation of residential hot water distribution systems by numerical simulation," Final report, Building Technology Center, Oak Ridge National Laboratory, Oak Ridge, Tennessee

9 University of Wisconsin-Madison, Solar Energy Lab, http://sel.me.wisc.edu/TRNSYS/Default.htm

10 NAHB Research Center, Inc., 2002, "Domestic hot water system modeling for the design of energy efficient systems,"

- Water and copper pipe temperature is a function of time only for a period following the time an outlet (faucet or shower) is deactivated.
- Insulation has no thermal capacity.
- Convective heat transfer coefficient on the air side of the at the external surface is independent of temperature and time.
- Heat conduction in the water and tube in the axial direction is negligible.
- Water flow is assumed to be fully developed.

Simplified governing equation:

HWDS On

$$\dot{m}C_{p,w}\frac{\partial T}{\partial x}+\left[\left(\rho_w C_{p,w}A_w\right)+\left(\rho_p C_{p,p}A_p\right)\right]\frac{\partial T}{\partial \tau}+UPT=UPT_\infty \qquad (3\text{-}1)$$

where

$\dot{m}$ = Water flow rate [kg/s]
$C_{p,w}$ = Water specific heat [J/kg.K]
$C_{p,p}$ = Pipe specific heat [J/kg.K]
T = Pipe and water temperature [^oC] = f(x,t)
T_∞ = Surrounding air temperature where a pipe is located [^oC]
ρ_w = Water density [kg/m^3]
ρ_p = Pipe density [kg/m^3]
A_w = Water flow area [m^2]
A_p = Pipe cross section area [m^2]
x = Pipe distance from hot water source [m]
τ = Time [s]
U = Overall heat transfer coefficient [W/m^2.K]

$$U=\frac{1}{\frac{1}{h_o}+\sum\frac{t_j}{k_j}} \qquad (3\text{-}2)$$

where

h_o = Heat transfer coefficient at the exterior pipe surface [W/m^2.K]
t_i = Thickness at i-th layer of a pipe [m]
k_i = Thermal conductivity at i-th layer of a pipe [W/m.K]

Let

$$a_m=\dot{m}C_{p,w}$$

$$a_\rho=\left(\rho_w C_{p,w}A_w\right)+\left(\rho_p C_{p,p}A_p\right)$$

Boundary condition:

$T(0,t) = T_{inlet}$ [^oC]

Initial condition:

$T(x,0) = T_a$ [^oC]

Numerical solution

Since the governing equation is a partial differential equation with respect to distance and time, the equation may be solved numerically using the following finite difference method:

$$T_{i,c} = \frac{a_m * T_{i-1,c} / \Delta x + a_\rho * T_{i,p} / \Delta \tau + UPT_\infty}{a_m / \Delta x + a_\rho / \Delta \tau + UP} \qquad (3\text{-}3)$$

where

$T_{i,c}$ = Water temperature at ith node and current time step

$T_{i,p}$ = Water temperature at ith node and previous time step

Δx = The distance between ith and (i+1)th node (L/200 is used in numerical solution)

$\Delta \tau$ = The time difference between previous time step and current time step

Analytical solution

The Laplace transform was used to solve the first order partial differential equation. The temperature distribution in a pipe is expressed below:

$$T(x,t) = T_\infty + \left[\left(T_{inlet} - T_\infty\right) * \exp\left(-\frac{UPx}{(\dot{m}C_p)_w} \right) - \left(T_a - T_\infty\right) * \exp\left(-\frac{UPt}{\left(\rho_w C_{p,w} A_w\right) + \left(\rho_p C_{p,p} A_p\right)} \right) \right] *$$
$$u\left(t - \frac{\left(\rho_w C_{p,w} A_w\right) + \left(\rho_p C_{p,p} A_p\right)}{(\dot{m}C_p)_w} x \right) + \left(T_a - T_\infty\right) * \exp\left(-\frac{UPt}{\left(\rho_w C_{p,w} A_w\right) + \left(\rho_p C_{p,p} A_p\right)} \right) \qquad (3\text{-}4)$$

where

u(t) is a unit step function and may be written as

$$u = \begin{cases} = 0 & when \;\; t < \dfrac{\left(\rho_w C_{p,w} A_w\right) + \left(\rho_p C_{p,p} A_p\right)}{(\dot{m} C_p)_w} x \\ = 1 & when \;\; t > \dfrac{\left(\rho_w C_{p,w} A_w\right) + \left(\rho_p C_{p,p} A_p\right)}{(\dot{m} C_p)_w} x \end{cases}$$

Heat losses

$$Q_{loss,on} = \int_0^{t_{on}} \int_0^L \left[UP(T(x,\tau) - T_\infty)dx\right] d\tau = \int_0^{t_{init}} \int_0^L \left[UP(T(x,\tau) - T_\infty)dx\right] d\tau + \int_{t_{init}}^{t_{on}} \int_0^L \left[UP(T(x,\tau) - T_\infty)dx\right] d\tau$$

(3-5)

where

$$\int_0^{t_{init}} \int_0^L \left[UP(T(x,\tau) - T_\infty)dx\right] d\tau = \int_0^{t_{init}} \int_0^{\frac{a_m}{a_\rho}\tau} \left[UP(T(x,\tau) - T_\infty)dx\right] d\tau + \int_0^{t_{init}} \int_{\frac{a_m}{a_\rho}\tau}^L \left[UP(T(x,\tau) - T_\infty)dx\right] d\tau =$$

$$L * a_\rho * (T_a - T_\infty) * \left[1 - e^{-\frac{UP}{a_\rho} t_{init}}\right] + a_m * (T_{inlet} - T_\infty) * \left[t_{init} + \frac{a_\rho}{UP}\left(e^{-\frac{UP}{a_\rho} t_{init}} - 1\right)\right] +$$

$$(T_a - T_\infty) * \frac{a_m a_\rho}{UP}\left[e^{-\frac{UP}{a_\rho} t_{init}} * \left(\frac{UP}{a_\rho} t_{init} + 1\right) - 1\right]$$

(3-6)

$$\int_{t_{init}}^{t_{on}} \int_0^L \left[UP(T(x,\tau) - T_\infty)dx\right] d\tau = UP(t_{on} - t_{init}) \int_0^L (T_{inlet} - T_\infty) * \exp\left(-\frac{UPx}{\dot{m} C_p}\right) dx =$$

$$\dot{m} C_p (t_{on} - t_{init}) * (T_{inlet} - T_\infty) * \left(1 - \exp\left(-\frac{UPL}{\dot{m} C_p}\right)\right)$$

(3-77)

$$t_{init} = \frac{\left(\rho_w C_{p,w} A_w\right) + \left(\rho_p C_{p,p} A_p\right)}{(\dot{m} C_p)_w} x$$

(3-8)

<u>HWDS Off</u>

Mass flow rate is set to zero.

$$\left[\left(\rho_w C_{p,w} A_w\right)+\left(\rho_p C_{p,p} A_p\right)\right]\frac{\partial T}{\partial \tau}+UPT=UPT_\infty \qquad (3\text{-}9)$$

Initial condition

$T(0) = T_{init}$

Analytical solution

$$T(t)=(T_{init}-T_\infty)*\exp\left(-\frac{UPt}{\left(\rho_w C_{p,w} A_w\right)+\left(\rho_p C_{p,p} A_p\right)}\right) \qquad (3\text{-}10)$$

Heat losses

$$Q_{heat,off}=\left[\left(\rho_w C_{p,w} A_w\right)+\left(\rho_p C_{p,p} A_p\right)\right]*(T_{init}-T_\infty)*\left[1-\exp\left(-\frac{UPt}{\left(\rho_w C_{p,w} A_w\right)+\left(\rho_p C_{p,p} A_p\right)}\right)\right]*L \qquad (3\text{-}11)$$

3.3 Model validation

Copper pipe at D = 0.75 in
Pipe length = 77 ft
Inlet temperature = 131 F
Initial temperature = 110 F
Water flow rate = 2 gpm
Ambient temperature = 90 F
On time: 180 sec
Off time: 5 minutes

Figure 1 shows measured11 and predicted temperatures at the shower outlet before and during shower activation. The predicted temperatures were obtained from both numerical and exact solutions. While the numerical approach is only an approximation, it is very close to the measured data. The accuracy is dependent on magnitude of Δx and $\Delta\tau$. Note that the exact solution shows temperature jump at time = 63 seconds instead of a slow temperature change. This occurs because, due to the simplifying assumptions, a unit function is used. In this case, the numerical approach provides the better solution for temperature prediction. From a total energy loss perspective, integrating temperature with respect to time, both solutions provide similar results. The difference of the integrated areas between 40 and 63 seconds is equal to the difference of the integrated areas between 63 and 78 seconds. Hence, predicted energy losses are virtually identical for both the numerical and exact solutions. The energy loss during the shower on time (180 seconds) is 6246.633 J from the numerical solution, and 6246.603 J from the exact solution.

11 Data measured at a Brevard County Florida residence in April 2005. See Appendix B for data.

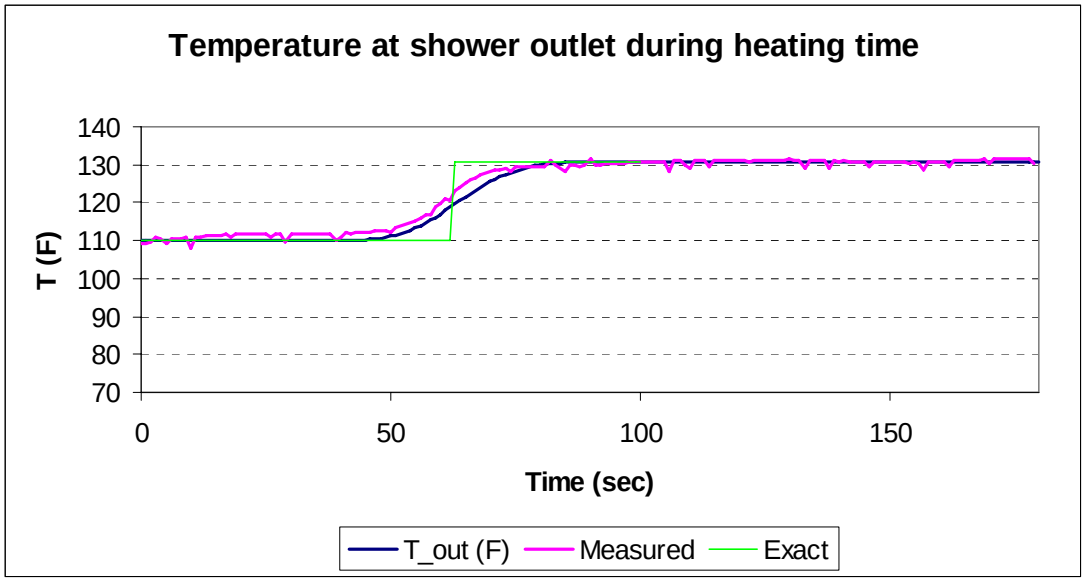

Figure 1. Temperature comparison between measurement and prediction at shower outlet during heating time

Figure 2 plots the temperature comparison between measurement and prediction at the shower outlet after the shower is turned off. Since it is easy to obtain an exact solution, no numerical approach is needed. As shown in the figure, the data match quite well.

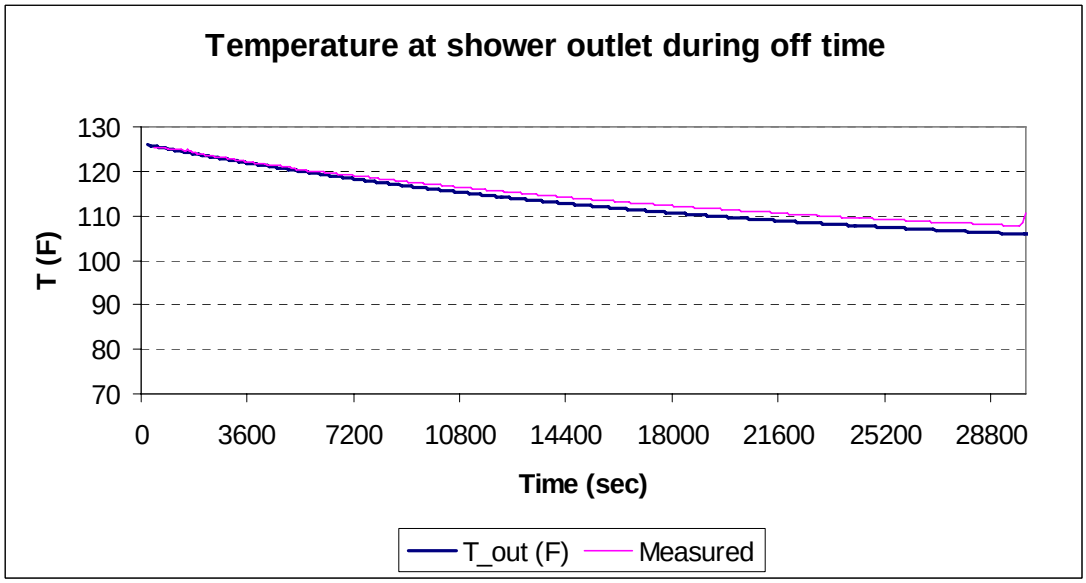

Figure 2. Temperature comparison between measurement and prediction at shower outlet during heating off period

3.4 Model application

Even though simplifying assumptions are used for the governing equations, the above section shows that the model can predict the temperature distribution and energy losses very well. The next step is to integrate the model into a whole building simulation program, so that energy losses from HWDS becomes a part of energy uses in a whole

building. DOE-2 is selected as a whole building simulation program. The input function is named as DHWLOADS, and is called in the Zone section before zone calculation is performed in the system computation. The required input values are:

- Tank size [gal]
- Tank water set temperature [^{o}F]
- Pipe diameter [in]
- Pipe length [ft]
- Water flow rate [gal/min]
- Thermal resistance of pipe insulation [R]
- Copper pipe thickness [in]
- Zone temperature where the pipes are located
- Water use schedule

Calculation procedure uses exact solutions for both heating on and off periods in the following steps:

1. Check if the water heater is on or off based on given water heater operation schedule
2. If water heater is off, calculate pipe heat losses based on Eq. (11)
3. If water heater is on, calculate pipe heat losses based on Eq. (5) during on time fraction, then calculate pipe heat losses based on Eq. (11) during off time fraction
4. Save pipe temperature for next time step use
5. Add the pipe energy loss into DHWKW, variable for hot water heater energy use

Note: When hot water heater is on during the whole hour, the pipe heat losses are calculated for an hour. When the hot water heater is on for a fraction of the hour, it is assumed that the heater is on from the beginning of the hour, and off in the rest of the hour.

```
FUNCTION NAME = DHWLOADS ..
  ASSIGN  IHR=IHR  IDAY=IDAY  IMO=IMO  INILZE=INILZE
          DHWSIZ = DHWSIZ $SIZE OF DHW TANK$
          DHWGAL = DHWGAL $GAL/MIN WITH SKED= 60 GALS/DAY; VALUE SHOULD
BE 30 + 10 * BROOMS$
          DHWTMP = DHWTMP $DHW WATER SET TEMPERATURE$
          DHWD = DHWD $ HWD Pipe diameter [in]
          DHWL = DHWL $ HWD pipe length [ft]
          DHWQ = DHWQ $ HWD water flow rate [gpm]
          DHWR = DHWR $ Thermal resistance of pipe insulation
          DHWT = DHWT $ Copper pipe thickness [in]
          PI = 3.1415927
          DTP=XXX50  $ Undocumented trick to save pipe temperature $
          DHWLOSS = XXX51 $ Hot water heater loss
          TLIVIN = XXX24  $ Living Zone temp $
          QS = QS               $ Sensible cooling loads $
          QL = QL  ..              $ heating coil loads $
  CALCULATE ..
C  WATER HEATING CONSUMPTION BY HOUR IS DETERMINED
```

```
          DHWFR= .2
          IF (IHR .EQ. 2) DHWFR=.1
          IF (IHR .EQ. 3) DHWFR=.0
          IF (IHR .EQ. 4) DHWFR=.0
          IF (IHR .EQ. 5) DHWFR=.0
          IF (IHR .EQ. 6) DHWFR=.2
          IF (IHR .EQ. 7) DHWFR=.4
          IF (IHR .EQ. 8) DHWFR=1.0
          IF (IHR .EQ. 9) DHWFR=.8
          IF (IHR .EQ. 10) DHWFR=.82
          IF (IHR .EQ. 11) DHWFR=.8
          IF (IHR .EQ. 12) DHWFR=.72
          IF (IHR .EQ. 13) DHWFR=.64
          IF (IHR .EQ. 14) DHWFR=.56
          IF (IHR .EQ. 15) DHWFR=.48
          IF (IHR .EQ. 16) DHWFR=.44
          IF (IHR .EQ. 17) DHWFR=.52
          IF (IHR .EQ. 18) DHWFR=.64
          IF (IHR .EQ. 19) DHWFR=.76
          IF (IHR .EQ. 20) DHWFR=.84
          IF (IHR .EQ. 21) DHWFR=.76
          IF (IHR .EQ. 22) DHWFR=.68
          IF (IHR .EQ. 23) DHWFR=.6
          IF (IHR .EQ. 24) DHWFR=.52

C       Start to calculate pipe losses of hot water distribution system
          h_init = 0.0
          h_use = 0.0
          H_sby = 0.0
          t_init = 0.0
          t_on = 0.0
          DHWLOSS = 0.0
C          GOTO 101
          DMCP = DHWQ*0.0000631*1000.0*4180.0
          DArea = (DHWD/2.0*0.0254)*(DHWD/2.0*0.0254)*3.1415927
          DRCpA = DArea*1000.0*4180.0+DHWD*0.0254*PI*DHWT*0.0254*390*8910
          DUP = 1.0/(1.0/7.5+DHWR*0.176)*DHWD*0.0254*PI
          DPL = DHWL*0.3048
          IF (IMO .eq. 1 .and. IDAY .eq. 1 .AND. IHR .EQ. 1) DTP = 25.0
          IF (IMO .eq. 1 .and. IDAY .eq. 1 .AND. IHR .EQ. 1) TLIVIN=23.8
          DTIN = (DHWTMP-32)/1.8
          DTAM = (TLIVIN-32)/1.8
          T_s = DTP
          IF (DHWFR .eq. 0) GOTO 85
C         Initial stage
          t_init = DRCpA/DMCP*DPL
          h1 = DPL*DRCpA*(DTP-DTAM)*(1.0-exp(-DUP/DRCpA*t_init))
          h2 = DMCP*(DTIN-DTAM)*(t_init+DRCpA/DUP*(EXP(-DUP/DRCpA*t_init)
     &         -1.0))
          h3 = (DTP-DTAM)*DRCPA*DMCP/DUP*(EXP(-DUP/DRCPA*t_init)*
     &         (DUP/DRCPA*t_init+1.0)-1.0)
          h_init = h1+h2+h3
C         Use stage
          t_on = DHWFR*DHWSIZ/DHWQ*60
          h_use = DMCP*t_on*(DTIN-DTAM)*(1.0-EXP(-DUP*DPL/DMCP))
          t_s = (DTAM+(DTIN-DTAM)*EXP(-DUP*DPL/DMCP))
85        T_off = 3600-t_init-t_on
```

```
        if (T_off .LE. 0) goto 101
        DTP = (T_s-DTAM)*exp(-DUP/DRCPA*T_off)+DTAM
        H_sby = DRCPA*(T_s-DTAM)*(1.0-exp(-DUP/DRCPA*T_off))*DPL
        H_loss = (H_init+h_use+h_sby)/3600*3.4123
        DHWLOSS = (H_init+h_use)/3600/1000
86      QS = QS+H_loss
101     CONTINUE

      END
END-FUNCTION ..
```

The following table lists annual simulation results in a home in Miami with and without HWDS losses, extracted from Report BEPS in units of MBtu. The domestic hot water heater energy use increases 2.4%, and whole building annual energy use increases 1.6% due to HWDS losses.

Table 1: Building energy performance summary with and without HDWS losses

Categary	HWDS MBtu	No HWDS MBtu
AREA LIGHTS	4	4
MISC EQUIPMT	13.6	13.6
SPACE HEAT	0	0
SPACE COOL	3.8	3.6
VENT FANS	1	1
DOMHOT WATER	8.6	8.4
TOTAL	31.1	30.6

Outside of this contract, FSEC will incorporate this algorithm for hot water distribution in its software products.

3.5 Simplified Hot Water Distribution Modeling Conclusions

A simplified model to calculate HWDS energy losses, including thermal capacity impact, was developed. The model was validated against limited measured data and was successfully integrated into a whole building simulation program to calculate impact of HWDS energy losses on whole building energy use.

Although the code is written as an input function of DOE-2, the input function can be used as a general function to calculate HWDS losses, as long as required inputs are available.

Due to limitation of DOE-2, the input function is only able to simulate straight piping of the same size. However, the governing equations may be easily integrated into a network model to calculate heat losses in a realistic HWDS.

4.0 Ventilation models

4.1 Expanded ventilation capabilities

Recently, outside of this contract, FSEC expanded EnergyGauge ventilation control capabilities by adding a max-time damper control for ventilation systems. This was implemented through a function that runs the fan between a specified minimum and maximum runtime. Exact implementation of this will depend on the simulation program used. FSEC has added a private function to DOE-2 and the algorithm is incorporated into that function. Building America teams can now choose the following mechanical ventilation strategies:

- No mechanical ventilation provided
- Supply air fan
- Exhaust air fan
- Both supply and exhaust air fan (Fully or partially balanced)
- Enthalpy recovery ventilation system
- Runtime ventilation where ventilation air is provided only when heating and cooling systems run (supply vent using the air handler unit)
- Runtime ventilation with a required minimum where the HVAC fan runs for a minimum amount of time each hour
- Runtime ventilation where the outside air damper will close if the air handler system has run a set amount of time during the hour
- A system that has a required minimum runtime and a closure for the outside air damper after a maximum amount of time run that hour
- A system that provides no outdoor ventilation air but does provide a set ventilation fan power (this is primarily for some reference building energy use rule sets).

4.2 Reporting Ventilation Fan Energy

DOE-2 reports the fan energy in report SS-L. This SS-L report allows for separate reporting of ventilation fan energy during non-heating and non-cooling hours. In order to process scoring requirements that consider the energy use of mechanical fans (HERS 2006 for instance), the ventilation fan energy used during heating and cooling hours is proportioned to heating and cooling in accordance with those energy uses. For allocation purposes, the fan energy used during non-heating and non-cooling hours, which DOE2 reports on the SS-L report, is added to the total by the proportion of heating and cooling fan energy used that month. If no heating or cooling fan energy was used that month then 50% is added to each.

Listing fan energy use separately is a challenge. During runtime ventilation fan energy is not separated from standard air handler use even though the act of bringing in extra air may increase the air handler runtime. For the benchmark and other reference home buildings (e.g., HERS 2006 reference), the annual mechanical vent fan energy use is

calculated according to formulas and is divided into heating and cooling portions based on the ratio of heating to cooling energy use. hours.

Greater accuracy in separating out fan power can be obtained from EnergyPlus or other tools that can report energy use within smaller time steps. This becomes important for the runtime ventilation with minimum or maximum controls to separate out the time steps when the system would have run anyhow and the time steps where it does not.

4.3 Ventilation fan energy use example

The following example results are for a St. Louis home that would score 30% on the Building America benchmark analysis. The 2040 square-foot home is modeled with an ach50 of 4.0 and a SEER 14/ HSPF 8.5 heat pump. More details of the home are described in another report.[12] The following mechanical ventilation options produce the following results using EnergyGauge USA, version 2.5, release 10 (pre-release developer's version) and the *Calculate > Annual Simulation* menu option with the following key parameters set to minimize influences on the results:

- no natural ventilation allowed (no opening windows for passive cooling as the EnergyGauge program shuts off all mechanical ventilation during times when algorithms indicate conditions are favorable for opening windows)
- auto-sizing set to off – all results based on Cooling system size of 28.9 kBtu/h and Heating system size of 49.7 kBtu/h.

Results will be different for rated, proposed code and prototype buildings due to rules that alter the amount of ventilation to assure that specified rule set standards are met.

Table 2. Cooling energy use (kWh) variation with mechanical ventilation fan strategy

Mechanical Vent Method: 50 cfm rate entered for all	Cooling	Cooling and Mech. Vent Fan	Mech Vent Fan when cooling system off*	Total Cooling	% Increase from no vent
None	1913	349	0	2262	0
Supply Vent, 20 W continuous	1996	424	11	2431	7.5%
Exhaust Vent, 20 W continuous	1979	421	11	2411	6.6%
Balanced Vent, 40 W continuous	2046	490	21	2557	13.0%
0.6 effective ERV, 40 W continuous	1981	484	23	2488	10.0%
Runtime Vent	1947	355	0	2302	1.8%
Runtime Vent w/25% min. runtime	1967	510	94	2571	13.7%
Runtime vent w/outside damper off at 25% max. runtime	1920	351	0	2271	0.40%
Runtime Vent w/25% min. and 25% max.	1940	515	93	2548	12.6%

*Represents added energy use during hours when there is no cooling or heating as proportioned to cooling

[12] Fairey, Philip, Carlos Colon, Eric Martin, Subrato Chandra, "**Comparing Apples, Oranges and Grapefruit:** An Analysis of Current Building Energy Analysis Standards for Building America, Home Energy Ratings and the 2006 International Energy Conservation Code," FSEC_CR_1650-06, September, 2006.

Table 3. Heating energy use (kWh) variation with mechanical ventilation fan strategy

Mechanical Vent Method: 50 cfm rate entered for all	Heating	Heating and Mech. Vent Fan	Mech Vent Fan when heating system off*	Total Heating	% Increase from no vent
None	5608	826	0	6434	0
Supply Vent,20 W	5971	951	25	6947	8.0%
Exhaust Vent, 20 W continuous	6003	955	25	6983	8.5%
Balanced Vent, 40 W continuous	6733	1124	49	7906	22.9%
0. 6 effective ERV, 40 W continuous	6025	1033	49	7107	10.5%
Runtime Vent	5634	824	0	6462	0.44%
Runtime Vent w/25% min. runtime	5608	937	169	6714	4.4%
Runtime vent w/outside damper off at 25% max. runtime	5624	828	0	6452	0.28%
Runtime Vent w/25% min. and 25% max.	5598	936	169	9703	4.2%

*Represents added energy use during hours when there is no cooling or heating as proportioned to heating

Some explanation may help one understand the results shown in Tables 2 and 3.

4.3.1 Continuous Ventilation Systems

The exhaust vent option uses slightly more energy for heating, but slightly less for cooling due to the heat of the fan being added to the space for the supply fan but not for the exhaust fan.

Balanced air flow results in larger ventilation rates due to the governing equation (4-1) for combining forced and natural ventilation.

Eq. 4 -1 $$Q_{total} = (Q_{nat}^2 + Q_{unbal}^2)^{0.5} + Q_{Bal}$$

where Q represents volume of air flow (cfm or m/s).

We also assumed balanced flow required twice the fan power of unbalanced flow (40W vs. 20W). Even when a 60% enthalpy recovery ventilator (ERV) is added, the energy use is greater than for an unbalanced simple ventilation system.

4.3.2 Runtime Ventilation Systems

The runtime vent method uses the heating and cooling system fan and a purposeful, ducted return leak "hole" with a damper to bring in outside air when the system runs. Without any other controls, it only brings in fresh air only during periods when heating or cooling requires the air handler to run. For the St. Louis home, the runtime vent option only slightly increased heating and cooling energy use. Considering that we were only adding 50 cfm when the system runs this is not surprising. For the St. Louis climate, the home's mechanical systems were only on 21.6% of the time. Thus, on average for the

year, the home only was mechanically ventilated at an equivalent of 10.8 cfm (39.2 cfm less than the continuous vent runs) and the net effect when **combined** with the envelope ach50 leakage of 4.0 is very small. How small? Computing the difference between straight natural infiltration and the total from the runtime ventilation run requires looking at the difference between the flow calculated from equation 4-1 and what would have otherwise occurred.

Eq. 4-2 $Q_{difference} = Q_{total} - Q_{nat}$

Figure 3 represents the hourly Q_{nat} and $Q_{difference}$ for the runtime ventilation case. The average $Q_{difference}$ value is 2.6 cfm. Thus, runtime vent is hardly any different, on an annual basis, than no mechanical venting. Peak summer hours for this case were as high as 26 cfm and thus for some select hours the mechanical ventilation may make a significant difference but not on an annualized basis.

Runtime ventilation is highly dependent on system size. The system size entered (49.7 kBtu/h for winter and 28.9 kBtu/h for cooling) yielded very low winter runtimes as shown on the top of Figure 3.

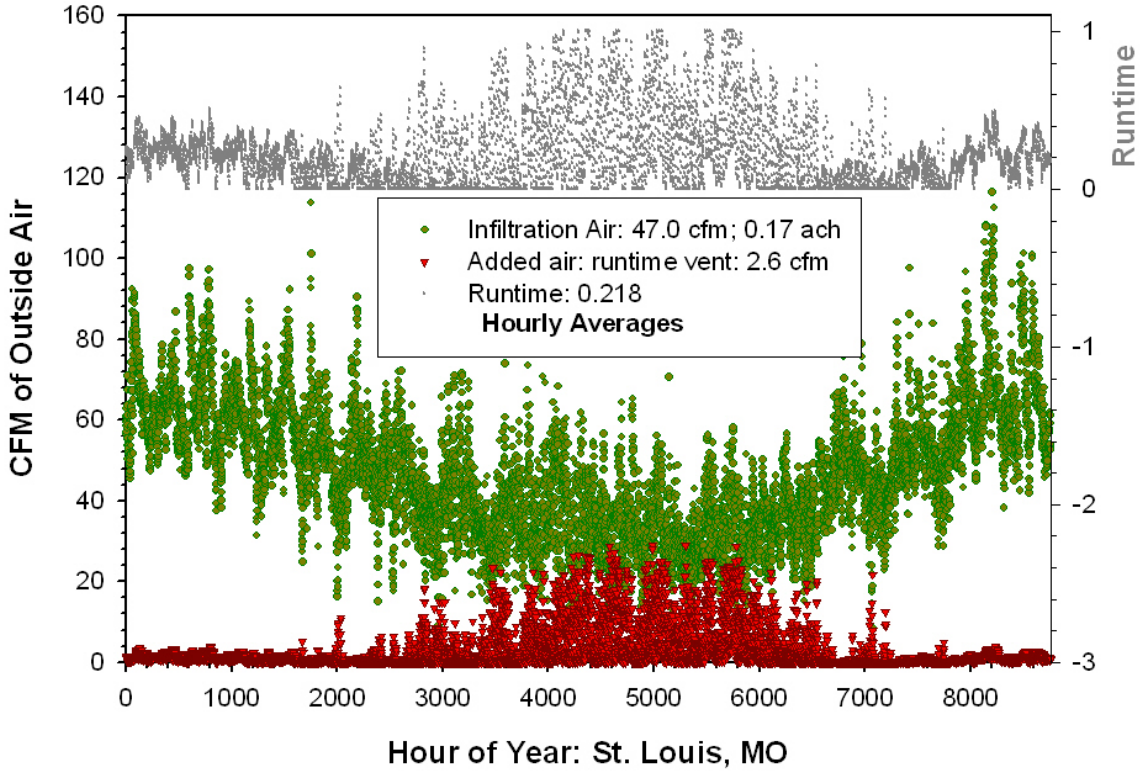

Figure 3. Hourly natural and added ventilation rates for runtime vent case. Inputs were 50 cfm mechanical and 4 ach50 leakage (natural). Natural infiltration is adjusted hourly by DOE2 based on natural driving forces (e.g., wind speed).

Requiring the ventilation system to run at least 25% of each hour increases heating by 4 % and cooling by 14% compared to the no-vent scenario. On the other hand, if the runtime vent is limited with a damper to be no greater than 25% of the hour, the model predicts almost no difference in cooling or heating energy use. This is as expected because the system will supply even less outside air than the simple runtime vent case shown in Figure 3, where for some hours it is adding ventilation air for much more than 25% of the hour. Finally, a sophisticated controller that maintains exactly 25% minimum and maximum runtime each hour results in a 4% increase in heating and a 13% increase in cooling energy use compared to no venting, , or slightly less energy penalty than the simpler 25% minimum runtime.

4.3.3. Fan Energy Use Explains Overall Energy Use Changes

Examining the breakout between actual cooling/heating and the fan energy use, it is apparent that most of the added energy is from the fan. The percentage increase in cooling for runtime vent with minimum is much higher than the heating percentage simply because the extra fan energy is a higher percentage of the total cooling. Actual cooling load is only slightly larger, not surprising as buildings require cooling many times when it is more comfortable outside. This occurs due to internal and solar gains creating cooling loads but reducing heating loads. Additionally, considering the fan motor adds extra internal load (166 kWh during cooling hours for the 25% fixed runtime case), it can explain all the difference in the column labeled cooling energy in Table 2 (27 kWh difference in the 25% fixed runtime case).

4.3.4. Fan Heat Energy is Extra Load

The heating value column in Table 2 is slightly misleading as the extra fan runtime also provides heat from its motor. Thus, the 25% fixed runtime case shows less heating (excluding fans) than the no vent case, but the software models the extra 279 kWh of fan energy as heat which in this case, with minimal added outside air, more than makes up for the added heating load due to infiltration.

5.0 Conclusions

FSEC has added capabilities for modelers to more accurately and more readily perform Building America energy analysis. The BA Toolkit allows programmers to obtain Benchmark characteristics easily. The ToolKit has been verified against the NREL spreadsheet tool. The spreadsheet tool proved an excellent method for validating the software as well as a good source for programming the algorithms for the ToolKit.

The simple hot water distribution algorithm FSEC developed provides a method to more accurately account for losses, and FSEC has demonstrated a “typical” straight piping loss in a Miami home may represent an increase of 2.4 % of the hot water energy use and also a slight increase in cooling energy.

Ventilation energy use methodologies have been expanded. Results from example runs indicate that fresh air provided by systems as well as energy use due to ventilation energy can vary significantly depending on the system control characteristics, even while sizing the ventilation system for the same amount of outside air *when* venting. Simple runtime vent systems may only bring in air 20% to 25% of the time on an annual average compared to continuous vent systems. The actual amount of additional air brought in relative to a natural infiltration only case can be very small as the unbalanced supply air is added in quadrature with natural ventilation. In a simple runtime ventilation scheme the model projects an annual average of only 2.6 cfm for a home with a 50 cfm runtime vent rate and 4 ach50 envelope leakage.

Assuring that the runtime vent system operates 25% of every hour results in a small energy penalty (4% and 13% of heating and cooling energy, respectively, in the St. Louis example). Much of this penalty is due to the energy use of the fan. Balanced ventilation strategies result in more outside air and typically require more fan energy use than supply or exhaust systems. Simulation results for St. Louis predict that even with 60% recovery, enthalpy ventilation systems may result in greater energy use than supply or exhaust ventilation systems that take half the fan energy.

Appendix A. Building America Benchmark Toolkit Software Description Details

Classes

- BA_BenchmarkEnvelope
- BA_BenchmarkAppliances
- BA_BenchmarkEquipment
- BA_MiscCalc

BA_BenchmarkEnvelope

Description: The class contains methods to calculate the envelope parameters of the Building America benchmark home.

Method Summary	
	BA_Benchmark2006EnvelopeMaster(Htg_Deg_Days As Single, Floor_Area As Single, Wall_Area_Gross As Single, Wall_Area_Belowgrade As Single, Wall_Area_Common As Single, Weather_Factor As Single, BasementFloorArea As Single, Basement_Gross_Wall_Area_Above_Grade As Single, Attach As Boolean, BasementConditioned As Boolean)
Double	GetInsRValue(FF As Double, Uo As Double, StudR As Double, RestR As Double, GetInsRError As Boolean)
	Ref_DoorU( )
Single	Ref_DoorArea( )
	Ref_Window_Area(Floor_Area As Single, Wall_Area_Common As Single, Wall_Area_Gross As Single, Wall_Area_Belowgrade As Single, BasementConditionedFloorArea As Single, Basement_Gross_Wall_Area_Above_Grade As Single, Attach As Boolean, BasementConditioned As Boolean)
	Ref_Wall_Ucalc(Htg_Deg_Days As Single, Attach As Boolean)
	Ref_Window_Ucalc(Htg_Deg_Days As Single)
	Ref_Floor_Ucalc(Htg_Deg_Days As Single)
	Ref_SlabFloor_Ucalc(Htg_Deg_Days As Single)
	Ref_RaisedFloor_Ucalc(Htg_Deg_Days As Single)
	Ref_Basement_Ucalc(Htg_Deg_Days As Single)
	Ref_Crawlspace_Ucalc(Htg_Deg_Days As Single)
	Ref_Ceiling_Ucalc(Htg_Deg_Days As Single)
Single	Ref_WallSolarAbsorptance( )

Single	Ref_WallEmittance( )
Single	Ref_RoofSolarAbsorptance( )
Single	Ref_RoofEmittance( )
	Ref_Framing_Fractions( )
	Ref_RoofInsulation( )
Single	Ref_Infiltration(Weather_Factor As Single)
	Ref_ShadingCoefficients(Htg_Deg_Days As Single)
	Ref_Windows( )
	Ref_Roof( )
	Final_U_Values( )
	ValueError(compname As String, compvalue As Single)
TBA_Benchmark EnvCalcOut	Get_Envelope_Outputs( )
Single	FurnitureMass( )
Single	PercentCarpet( )
Single	SlabInsulationDepth(Htg_Deg_Days As Single)
Single	BasementSpecificLeakageArea(Wall_Area_Belowgrade As Single, Weather_Factor As Single, Basement_Gross_Wall_Area_Above_Grade As Single)
Single	OverallSpecificLeakageArea(Floor_Area As Single, Wall_Area_Belowgrade As Single, Weather_Factor As Single, BasementConditionedFloorArea As Single, Basement_Gross_Wall_Area_Above_Grade As Single)
Boolean	HDDError(HDD As Single)
String	GetGlassType(Uvalue As Single)
	Mixed_Floor(Tile_Frac As Single, Hardwood_Frac As Single, Carpet_Frac As Single, TH As Single, Cond As Single, Dens As Single, SH As Single, MixedRvalue As Single)
Single	ConvertSHGCtoSC(SHGC As Single, Screen_Multiplier As Single)

Method Detail

BA_Benchmark2006EnvelopeMaster

BA_Benchmark2006EnvelopeMaster (Htg_Deg_Days As Single, Floor_Area As Single, Wall_Area_Gross As Single, Wall_Area_Belowgrade As Single, Wall_Area_Common As Single, Weather_Factor As Single, BasementFloorArea As Single, Basement_Gross_Wall_Area_Above_Grade As Single, Attach As Boolean, BasementConditioned As Boolean**)**

This method calls each envelope component routine.

Parameters:

Htg_Deg_Days - use NREL values from Calcs5 page of spreadsheet
Floor_Area - conditioned floor area of home (ft2)
Wall_Area_Gross - gross wall area of non-basement portion of home includes window and door areas within wall
Wall_Area_Belowgrade - wall area below grade (ft2)
Wall_Area_Common - for attached dwellings this is the wall area of the common wall (ft2)
Weather_Factor - based on the city
BasementFloorArea- the area of the basement (ft2)
Basement_Gross_Wall_Area_Above_Grade - the above-grade portion of the basement wall area (ft2)
Attach – true if house is considered an attached dwelling
BasementConditioned – true if home has part of basement conditioned

Returns:

None

Throws:

None

GetInsRValue

Double **GetInsRValue** (FF As Double, Uo As Double, StudR As Double, RestR As Double, GetInsRError As Boolean)

This method returns the Insulation R Value based on the envelope assembly (wall, roof, floor, etc.) parameters solving for parallel heat flow.

Parameters:

FF – Framing Factor of assembly (0 to 1.0)
Uo - Overall U value of assembly such as those returned from Benchmark envelope routines
StudR - The R value of the stud

RestR - The R value of all parts in common with both the framing and insulating portions of assembly

GetInsRError – True if an error is found in routine

Returns:

GetInsRValue

Throws:

None

Ref_DoorU

Ref_DoorU( )

This method

Parameters:

None

Returns:

None

Throws:

None

Ref_DoorArea

Single **Ref_DoorArea( )**

This method sets the door are of the home

Parameters:

None

Returns:

Reference door area

Throws:

None

Ref_Window_Area

Ref_Window_Area(Floor_Area As Single, Wall_Area_Common As Single, Wall_Area_Gross As Single, Wall_Area_Belowgrade As Single, BasementFloorArea As Single, Basement_Gross_Wall_Area_Above_Grade As Single, Attach As Boolean, BasementConditioned As Boolean)

This method calculates the window area, Fa, and F of the home

Parameters:

Htg_Deg_Days – Heating Degree Days base 65 F

Floor_Area - conditioned floor area of home (ft2)
Wall_Area_Common - for attached dwellings this is the wall area of the common wall (ft2)
Wall_Area_Gross - gross wall area of non-basement portion of home includes window and door areas within wall
Wall_Area_Belowgrade - wall area below grade (ft2)
BasementFloorArea- the area of the basement (ft2)
Basement_Gross_Wall_Area_Above_Grade - the above-grade portion of the basement wall area (ft2)
Attach – true if house is considered an attached dwelling
BasementConditioned – true if home has part of basement conditioned

Returns:

None

Throws:

None

Ref_Wall_Ucalc

Ref_Wall_Ucalc(Htg_Deg_Days As Single, Attach As Boolean)

This method calculates the overall U value of the wall assembly

Parameters:

Htg_Deg_Days - Heating Degree Days base 65 F
Attach - true if house is considered an attached dwelling

Returns:

None

Throws:

None

Ref_Window_Ucalc

Ref_Window_Ucalc(Htg_Deg_Days As Single)

This method calculates the overall U value fo the window assembly

Parameters:

Htg_Deg_Days - Heating Degree Days base 65 F

Returns:

None

Throws:

None

Ref_Floor_Ucalc

Ref_Floor_Ucalc(Htg_Deg_Days As Single)

This method calculates the overall U value of floor assemblies between conditioned and non-conditioned space

Parameters:
Htg_Deg_Days - Heating Degree Days base 65 F
Returns:
None
Throws:
None

Ref_SlabFloor_Ucalc

Ref_SlabFloor_Ucalc(Htg_Deg_Days As Single)

This method calculates the perimeter insulation U value

Parameters:
Htg_Deg_Days - Heating Degree Days base 65 F
Returns:
None
Throws:
None

Ref_RaisedFloor_Ucalc

Ref_RaisedFloor_Ucalc(Htg_Deg_Days As Single)

This method calculates the U value of the assembly

Parameters:
Htg_Deg_Days - Heating Degree Days base 65 F
Returns:
None
Throws:
None

Ref_Basement_Ucalc

Ref_Basement_Ucalc(Htg_Deg_Days As Single)

This method calculates the U value fo the basement walls

Parameters:
Htg_Deg_Days - Heating Degree Days base 65 F
Returns:
None
Throws:
None

Ref_Crawlspace_Ucalc

Ref_Crawlspace_Ucalc(Htg_Deg_Days As Single)

This method calculates the U value fo the crawlspace walls

Parameters:
Htg_Deg_Days - Heating Degree Days base 65 F
Returns:
None
Throws:
None

Ref_Ceiling_Ucalc

Ref_Ceiling_Ucalc(Htg_Deg_Days As Single)

This method calculates the U value of the ceiling assembly

Parameters:
Htg_Deg_Days - Heating Degree Days base 65 F
Returns:
None
Throws:
None

GetCeilingUvalue

Single **GetCeilingUvalue**(Htg_Deg_Days As Single)

This method calculates the

Parameters:
Htg_Deg_Days -
Returns:

Ceiling U Value

Throws:

None

Ref_WallSolarAbsorptance

Single **Ref_WallSolarAbsorptance**( )

This method calculates the exterior wall solar absorptance

Parameters:

None

Returns:

Reference Wall Solar Absorptance

Throws:

None

Ref_WallEmittance

Single **Ref_WallEmittance**( )

This method calculates the exterior wall emittance

Parameters:

None

Returns:

Reference Wall Emittance

Throws:

None

Ref_RoofSolarAbsorptance

Single **Ref_RoofSolarAbsorptance**( )

This method calculates the roof exterior solar absorptance

Parameters:

None

Returns:

Reference Roof Solar Absorptance

Throws:

None

Ref_RoofEmittance

Single **Ref_RoofEmittance**( )

This method calculates the exterior roof surface emissivity

Parameters:
None
Returns:
Reference Roof Emittance
Throws:
None

Ref_Framing_Fractions

Ref_Framing_Fractions( )

This method calculates the framing fractions for walls, floors, ceilings and roofs

Parameters:
None
Returns:
None
Throws:
None

Ref_RoofInsulation

Ref_RoofInsulation( )

This method calculates the insulation level of material adjacent to the roof

Parameters:
None
Returns:
None
Throws:
None

Ref_Infiltration

Single **Ref_Infiltration**(Weather_Factor As Single)

This method calculates the natural infiltration (leakage) of the home

Parameters:
Weather_Factor -
Returns:
Reference Infiltration
Throws:
None

Ref_ShadingCoefficients

Ref_ShadingCoefficients(Htg_Deg_Days As Single)

This method calculates the solar heat gain shading coefficient for the windows

Parameters:
Htg_Deg_Days - Heating Degree Days base 65 F
Returns:
None
Throws:
None

Ref_Windows

Ref_Windows( )

This method calculates related glass parameters:

```
BA_BenchmarkEnvCalcOut.Overhangs := 0;
BA_BenchmarkEnvCalcOut.Glass_Type : =
GetGlassType(BA_BenchmarkEnvCalcOut.UValuesAssembly.Window_);
BA_BenchmarkEnvCalcOut.Frame_Type := 'Vinyl';
BA_BenchmarkEnvCalcOut.Int_Shading := 'Drapes/blinds';
BA_BenchmarkEnvCalcOut.Screening := 'None';
```

Parameters:
None
Returns:
None
Throws:
None

Ref_Roof

Ref_Roof( )

This method sets the following roof outputs:

BA_BenchmarkEnvCalcOut.Roof_Deck_Insulation_Lvl := 0;
BA_BenchmarkEnvCalcOut.Roof_Color := 'White';
BA_BenchmarkEnvCalcOut.Roof_Material := 'Composition shingles';
BA_BenchmarkEnvCalcOut.Attic_Description := 'Full attic';
BA_BenchmarkEnvCalcOut.Roof_Config := 'Gable or shed';
BA_BenchmarkEnvCalcOut.Attic_Vent_Ratio := 1/300;

Parameters:
None
Returns:
None
Throws:
None

ValueError

ValueError(compname As String, compvalue As Single)

This method provides an error mesage

Parameters:
compname – The name of the component not meeting an error rule
compvalue - The value fo the component
Returns:
None
Throws:
None

Get_Envelope_Outputs

TBA_BenchmarkEnvCalcOut **Get_Envelope_Outputs**( )

This method returns all of the envelope characteristics for a given prototype home.

Parameters:
None
Returns:
Record consisting of outputs of Envelope Calculations

```
TBA_BenchmarkEnvCalcOut = record
  BA_Benchmark_Error: WordBool;
 Radiant_Barrier: WordBool;
 Attic_Vent_Ratio: Single;
 Ceiling_Framing_Fraction: Single;
```

```
        CrawlspaceWall_Framing_Fraction: Single;
        Door_Area: Single;
        Floor_Framing_Fraction: Single;
        Overhangs: Single;
        Roof_Deck_Insulation_Lvl: Single;
        Roof_Framing_Fraction: Single;
        SC_Cooling: Single;
        SC_Heating: Single;
        SHGC_Cooling: Single;
        SHGC_Heating: Single;
        SLA: Single;
        SolarAbsorptance_Roof: Single;
        SolarAbsorptance_Walls: Single;
        Wall_Framing_Fraction: Single;
        WindowArea: Single;
        WindowFrameArea: Single;
        SuperGrossWallArea: Single;
        TotalThermalBoundaryWallArea: Single;
        F_Attached: Single;
        FA: Single;
        Emittance_Roof: Single;
        Emittance_Walls: Single;
        SlabInsulationDepth: Single;
        PercentCarpet: Single;
        CarpetRValue: Single;
        FurnitureMass: Single;
        BasementSLA: Single;
        OVerallSLA: Single;
        Attic_Description: WideString;
        Frame_Type: WideString;
        Glass_Type: WideString;
        Int_Shading: WideString;
        Roof_Color: WideString;
        Roof_Config: WideString;
        Roof_Material: WideString;
        Screening: WideString;
        RValuesAssembly: TUandRvalues;
        RValuesInsulation: TUandRvalues;
        RStud: TUandRvalues;
        RRest: TUandRvalues;
        UValuesAssembly: TUandRvalues;
        UValuesInsulation: TUandRvalues;
Where
  TUandRvalues = record
  BasementCeil: Single;
  BasementWall: Single;
```

Ceil: Single;
CrawlspaceWall: Single;
CrawlspaceCeil: Single;
Door: Single;
FloorOverGarage: Single;
FloorRaised: Single;
Overall: Single;
Roof: Single;
SlabPerimeterUnheated: Single;
SlabPerimeterHeated: Single;
Wall: Single;
Window: Single;

Throws:
None

FurnitureMass( )

Single **FurnitureMass**( )

This method calculates the furniture mass for the building

Parameters:
None
Returns:
Furniture Mass
Throws:
None

PercentCarpet

Single **PercentCarpet**( )

This method calculates the percent carpet

Parameters:
None
Returns:
Carpet Percentage
Throws:
None

SlabInsulationDepth

Single **SlabInsulationDepth**(Htg_Deg_Days As Single)

This method calculates the depth of the perimeter insulation

Parameters:

Htg_Deg_Days - Heating Degree Days base 65 F

Returns:

Slab Insulation Depth

Throws:

None

BasementSpecificLeakageArea

Single **BasementSpecificLeakageArea**(Wall_Area_Belowgrade As Single, Weather_Factor As Single, Basement_Gross_Wall_Area_Above_Grade As Single)

This method calculates the specific leakage area of the basement

Parameters:

Wall_Area_Belowgrade - – the area (ft2) of basement wall that is below ground level

Weather_Factor – climate dependent wind/terrain parameter

Basement_Gross_Wall_Area_Above_Grade – the area (ft2) of basement wal that is above ground level

Returns:

Basement Specific Leakage Area

Throws:

None

OverallSpecificLeakageArea

Single **OverallSpecificLeakageArea**(Floor_Area As Single, Wall_Area_Belowgrade As Single, Weather_Factor As Single, BasementFloorArea As Single, Basement_Gross_Wall_Area_Above_Grade As Single)

This method calculates the home specific leakage area

Parameters:

Floor_Area - conditioned floor area of home (ft2)

Wall_Area_Belowgrade - wall area below grade (ft2)

Weather_Factor - based on the city

BasementFloorArea- the area of the basement (ft2)

Basement_Gross_Wall_Area_Above_Grade - the above-grade portion of the basement wall area (ft2)

Returns:

Overall Specific Leakage Area

Throws:

None

HDDError

Boolean **HDDError**(HDD As Single)

This method returns True if HDD is less than 0 or greater than 60,000

Parameters:

HDD – heating degree days (base 65 F)

Returns:

Throws:

None

GetGlassType

String **GetGlassType**(Uvalue As Single)

This method calculates the type of window that best matches the U-value

```
if Uvalue >= 0.9 then  Result := 'Single'
else if (Uvalue >= 0.57) and (Uvalue < 0.9) then Result := 'Double'
else if (Uvalue >= 0.45) and (Uvalue < 0.57) then Result := 'Low-E Double'
else if Uvalue < 0.45 then Result := 'Low-E Triple';
```

Parameters:

Uvalue – The U-value fo the overall window assembly

Returns:

Glass Type

Throws:

None

Mixed_Floor

Mixed_Floor(Tile_Frac As Single, Hardwood_Frac As Single, Carpet_Frac As Single, TH As Single, Cond As Single, Dens As Single, SH As Single, MixedRvalue As Single)

This method calculates the heat transfer properties of a single floor material that is calculated based on the fraction of floor areas and default properties for tile, sood and carpet.

Parameters:

Tile_Frac - fraction of floor area (0 to 1.0) that is tile

Hardwood_Frac – Fraction of floor area (0 to 1.0) that is wood
Carpet_Frac – Fraction of floor area (0 to 1.0) that is carpet

Returns:

TH – Mixed property thickness
Cond – Mixed property conductivity
Dens – Mixed floor property density
SH – Mixed floor property specific heat
MixedRvalue – The R-value of the mixed property

Throws:

None

ConvertSHGCtoSC

Single **ConvertSHGCtoSC**(SHGC As Single, Screen_Multiplier As Single)

This method converts SHGC to SC according to ASHRAE Fundamentals 1997, page 29.23, equation 38

Parameters:

SHGC – The solar heat gain coefficient for the window
Screen_Multiplier – Value (0 to 1) for any insect or other window screening

Returns:

Throws:

None

BA_BenchmarkAppliances

Description: The class calculates the energy use and hourly schedules of appliances and the sensible and latent internal loads form the appliances and occupants.

Method Summary	
	BA_Benchmark2006AppliancesMaster(Beds As Integer, FFA As Single, StateMultiplier As Single, DryerFuel As String, RangeFuel As String)
	Clotheswasher(Beds As Integer)
	Dishwasher(Beds As Integer)
	Dryer(Beds As Integer, DryerFuel As String)
	Lighting(FFA As Double)
	Misc(FFA As Double, StateMultiplier As Double)
	Occupancy(Beds As Integer)
	Range(RangeFuel As String)
	Refrigeration( )
Double	ConvertThermsToPropaneGal(Therms As Double)
TBA_BenchmarkAppliancesCalcOut	Get_Appliances_Outputs( )
	PlugInLighting(FFA As Double)

Method Detail

BA_Benchmark2006AppliancesMaster

BA_Benchmark2006AppliancesMaster (Beds As Integer, FFA As Single, StateMultiplier As Single, DryerFuel As String, RangeFuel As String)

This method calls each appliance component routine.

Parameters:

Beds – Number of Bedrooms

FFA – Finished Floor Area
StateMultiplier – Factor from NREL spreadsheet
DryerFuel – 'Electric' or 'Gas' or 'Propane'
RangeFuel – 'Electric' or 'Gas' or 'Propane'

Returns:

None

Throws:

None

Clotheswasher

Clotheswasher(Beds As Integer)

This method populates the TBA_BenchmarkAppliancesCalcOut.Cotheswasher_Info record.

Parameters:

Beds – Number of Bedrooms

Returns:

None

Throws:

None

Dishwasher

Dishwasher(Beds As Integer)

This method populates the TBA_BenchmarkAppliancesCalcOut.Dishwasher_Info record.

Parameters:

Beds **– Number of Bedrooms**

Returns:

None

Throws:

None

Dryer

Dryer(Beds As Integer, DryerFuel As String)

This method populates the TBA_BenchmarkAppliancesCalcOut.Dryer_Info record.

Parameters:
Beds **– Number of Bedrooms**
DryerFuel – 'Electric' or "Gas' or 'Propane'
Returns:
None
Throws:
None

Lighting

Lighting(FFA As Double)

This method populates the TBA_BenchmarkAppliancesCalcOut.Lighting_Info record.

Parameters:
FFA – Finished Floor Area
Returns:
None
Throws:
None

Misc

Misc(FFA As Double, StateMultiplier As Double)

This method populates the TBA_BenchmarkAppliancesCalcOut.Misc_Info record.

Parameters:
FFA – Finished Floor Area
StateMultiplier – Factor for determining miscellaneous use –use 1 if unknown, NREL spreadsheet has default values for each state
Returns:
None
Throws:
None

Occupancy

Occupancy(Beds As Integer)

This method populates the TBA_BenchmarkAppliancesCalcOut.Occupancy_Info record.

Parameters:
Beds – Number of Bedrooms
Returns:
None
Throws:
None

Range

Range(RangeFuel As String)

This method populates the TBA_BenchmarkAppliancesCalcOut.Range_Info record.

Parameters:
RangeFuel – 'Electric' or 'Gas' or 'Propane'
Returns:
None
Throws:
None

Refrigeration

Refrigeration( )

This method populates the TBA_BenchmarkAppliancesCalcOut.Refrigeration_Info record.

Parameters:
None
Returns:
None
Throws:
None

ConvertThermsToPropaneGal

Double **ConvertThermsToPropaneGal**(Therms As Double)

This method calls

Parameters:

Therms – The energy use in therms of the gas appliance

Returns:

Equivalent Gallons of Propane

Throws:

None

Get_Appliances_Outputs

TBA_BenchmarkAppliancesCalcOut **Get_Appliances_Outputs**( )

This method returns all of the appliance characteristics for a given prototype home.

Parameters:

None

Returns:

Record consisting of outputs of Appliance Calculations**:**

TBA_BenchmarkAppliancesCalcOut = record
Added_Elec_Annual_Use: Double;
Dishwasher_HWGallonsPerDay: Double;
Clotheswasher_HWGallonsPerDay: Double;
CeilingFan_Info: TBAAppliance_Info;
ClothesWasher_Info: TBAAppliance_Info;
Dishwasher_Info: TBAAppliance_Info;
Dryer_Info: TBAAppliance_Info;
Lighting_Info: TBAAppliance_Info;
Misc_Info: TBAAppliance_Info;
PoolPump_Info: TBAAppliance_Info;
Range_Info: TBAAppliance_Info;
Refrigeration_Info: TBAAppliance_Info;
Occupancy_Info: TBAAppliance_Info;
People: Single;
SensiblePerPerson: Single;
LatentPerPerson: Single;
PlugInLighting_Info: TBAAppliance_Info;

where
TBAAppliance_Info = record
Curr_Appliance_ID: Integer; alphabetical from 1 -9 (clotheswasher is 1, refrigeration is 9)
Curr_Annual_Use: Single; annual energy use
Curr_Peak_Demand: Single; largest hourly value of energy use
Curr_Percent_Released: Single; total percentage of energy released inside the home
Curr_Latent_Released: Single; percentage of total energy released as latent energy inside the home

Curr_Sensible_Released: Single; percentage of total energy released as sensible energy inside the home
Curr_Annual_Use_Type: WideString; units of annual energy use
Curr_Peak_Demand_Type: WideString; units of peak demand
Curr_Appliance_Type: WideString; 'Clotheswasher', or 'Dryer', or…,'Refrigeration'
Curr_Name: WideString; 'BA_Benchmark 2006'
Curr_Hour: array[1..24] of Single; fraction of peak demand for each hour

Throws:
None

PlugInLighting

PlugInLighting(FFA As Double)

This method populates the TBA_BenchmarkAppliancesCalcOut.PLugInLighting_Info record.

Parameters:
FFA – Finished Floor Area
Returns:
None
Throws:
None

BA_BenchmarkEquipment

Description: This class contains the methods to determine the benchmark home heating, cooling, air distribution, water heating and mechanical ventilation characteristics.

Method Summary	
	BA_Benchmark2006EquipmentMaster(ambientannualtemp As Single, baths As Single, FFA As Single, Cond_Area As Single, HeatCFM As Single, CoolCFM As Single, DHWdeliverytemp As Single, DHWsupplytemp As Single, HeatingCapacity As Single, CoolingCapacity As Single, Beds As Integer, Stories As Integer, Nreturns As Integer, NumHotWaterSystems As Integer, NumHeatingSystems As Integer, BasementConditioned As Boolean, CrawlSpaceConditioned As Boolean, PredominantFoundation As String, Heating_Fuel_Type As String, HotWater_Primary_Type As String, HotWater_Location As String, Heating_Type As String, HotWater_FuelType As String)
	Ducts(FFA As Single; Nreturns As Integer, Stories As Integer; BasementConditioned As Boolean, CrawlSpaceConditioned As Boolean, PredominantFoundation As String)
	Cooling ( )
	Heating(NumHeatingSystems As Integer, Heating_Fuel_Type As String, Heating_Type As String)
	WaterHeating(HotWater_Primary_Type As String, HotWater_FuelType As String, HotWater_Location As String, Beds As Integer, NumHotWaterSystems As Integer, baths As Single, ambientannualtemp As Single, DHWdeliverytemp As Single, DHWsupplytemp As Single)
	BAHeatingEquip(HeatTypeIn As String, FuelType As String, HeatEff As Single, HeatTypeOut As String)
Single	BACoolingEquip(CoolType As String, FuelType As String)
Single	BAWaterHeatingEF(WaterHeatingType As String, FuelType As String, WaterHeatingCap As Single)
Single	BAWaterHeatingRE(FuelType As String)
Single	BAWaterHeatingStorageVolume(Curr_Fuel_Type As String, Beds As Integer, baths As Single)
Single	BAWaterHeatingBurnerCapacity(Curr_Fuel_Type As String, Beds As Integer, baths As Single)
Single	BACoolingSetPoint ( )
Single	BAHeatingSetPoint ( )

Single	BAMechVentCFMFlow(CFA As Single, Beds As Integer)
Single	BAMechVentPower(CFA As Single, Beds As Integer)
	WaterHeatingGallonsPerDaybyMonth(Beds As Integer, ambientannualtemp As String, baths As String, deliverytemp As String, supplytemp As String)
TBA_Benchmark EquipCalcOut	Get_Equipment_Output ( )
	Set_AmbientMonthTemp(Jan As Single, Feb As Single, Mar As Single, Apr As Single, May As Single, Jun As Single, Jul As Single, Aug As Single, Sep As Single, Oct As Single, Nov As Single, Dec As Single)
Single	FindHighest(ArraySize As Integer)
Single	FindLowest(ArraySize As Integer)

Method Detail

BA_Benchmark2006EquipmentMaster

BA_Benchmark2006EquipmentMaster(ambientannualtemp As Single, baths As Single, FFA As Single, Cond_Area As Single, HeatCFM As Single, CoolCFM As Single, DHWdeliverytemp As Single, DHWsupplytemp As Single, HeatingCapacity As Single, CoolingCapacity As Single, Beds As Integer, Stories As Integer, Nreturns As Integer, NumHotWaterSystems As Integer, NumHeatingSystems As Integer, BasementConditioned As Boolean, CrawlSpaceConditioned As Boolean, PredominantFoundation As String, Heating_Fuel_Type As String, HotWater_Primary_Type As String, HotWater_Location As String, Heating_Type As String, HotWater_FuelType As String)

This method calls each equipment component routine.

Parameters:

ambientannualtemp – the annual average outdoor temperature in degrees F
baths – number of bathrooms
FFA – finished floor area
Cond_Area – the conditioned floor area

DHWdeliverytemp -
DHWsupplytemp -
Beds – Number of Bedrooms
Stories – Number f Stories
Nreturns – Number of returns
NumHotWaterSystems – Number of hot water systems
NumHeatingSystems – Number of heating systems
BasementConditioned – True if basement is conditioned
CrawlSpaceConditioned – True if crawlspace is conditioned
PredominantFoundation – 'Basement' or 'Crawlspace' or'Slab'
Heating_Fuel_Type – 'Electric' or 'Gas' or ''Propane' or 'Oil'
HotWater_Primary_Type - 'Electric' or 'Gas' or ''Propane' or 'Oil"
HotWater_Location -
Heating_Type -
HotWater_FuelType - 'Electric' or 'Gas' or ''Propane' or 'Oil"

Returns:

None

Throws:

None

Ducts

Ducts(FFA As Single; Nreturns As Integer, Stories As Integer; BasementConditioned As Boolean, CrawlSpaceConditioned As Boolean, PredominantFoundation As String)

This method calculates the Duct_info and air handler and other duct parameters.

Parameters:

FFA – Finished Floor Area
Nreturns – Number of returns
Stories – Number of stories
BasementConditioned – 'True' if conditioned
CrawlSpaceConditioned – 'True' if conditioned
PredominantFoundation – 'Basement' or 'Crawlspace' or'Slab'

Returns:

None

Throws:

None

Cooling

Cooling ( )

This method calculates the cooling equipment efficiency paremters

Parameters:
None
Returns:
None
Throws:
None

Heating

Heating(NumHeatingSystems As Integer, Heating_Fuel_Type As String, Heating_Type As String)

This method will base the heating system parameters based on the input of fuel and heating type. IF the number of heating systems is 0, it does notset the parameters.

Parameters:
NumHeatingSystems – The number of heating systems
Heating_Fuel_Type – The type of fuel used for the heating system Electric' or 'Gas' or 'Propane' or 'Oil'
Heating_Type **–** For non-electric fuel systems the efficiency will change if 'Heat pump' or 'Hydronic' is part of the heating type entered
Returns:
None
Throws:
None

WaterHeating

WaterHeating(HotWater_Primary_Type As String, HotWater_FuelType As String, HotWater_Location As String, Beds As Integer, NumHotWaterSystems As Integer, baths As Single, ambientannualtemp As Single, DHWdeliverytemp As Single, DHWsupplytemp As Single)

This method calculates the water heating parameters for the home

Parameters:
HotWater_Primary_Type -
HotWater_FuelType – 'Electric' or 'Natural Gas' or ''Propane' or 'Fuel Oil"
HotWater_Location – Location of the hot water system
Beds – Number of bedrooms
NumHotWaterSystems – Number of hot water systems
baths – Number of bathrooms

ambientannualtemp – Annual average outdoor temperature, degrees F
DHWdeliverytemp – Delivery temperature, degrees F
DHWsupplytemp – Supply temperature, degrees F

Returns:

None

Throws:

None

BAHeatingEquip

BAHeatingEquip(HeatTypeIn As String, FuelType As String, HeatEff As Single, HeatTypeOut As String)

This method calculates the heating type and heatingefficiency

Parameters:

HeatTypeIn - For non-electric fuel systems the efficiency will change if 'Heat pump' or 'Hydronic' is part of the heating type entered
FuelType - – The type of fuel used for the heating system Electric' or 'Gas' or 'Propane' or 'Oil'

Returns:

HeatEff – The efficiency of the hating system
HeatTypeOut – The type of system entered unless the fuel type is electric in which case it will return 'Electric Heat Pump'

Throws:

None

BAWaterHeatingEF

Single **BAWaterHeatingEF**(WaterHeatingType As String, FuelType As String, WaterHeatingCap As Single)

This method calculates the water heating efficiency

Parameters:

WaterHeatingType –
FuelType - Electric' or 'Natural Gas' or ''Propane' or 'Fuel Oil"
WaterHeatingCap – The capacity of the water heating tank

Returns:

Water Heating Efficiency

Throws:

None

BAWaterHeatingRE

Single **BAWaterHeatingRE**(FuelType As String)

This method calculates the recovery efficiency

Parameters:
FuelType - Electric' or 'Natural Gas' or ''Propane' or 'Fuel Oil"
Returns:
Water Heating Recovery Efficiency
Throws:
None

BAWaterHeatingStorageVolume

Single **BAWaterHeatingStorageVolume**(Curr_Fuel_Type As String, Beds As Integer, baths As Single)

This method

Parameters:
Curr_Fuel_Type – 'Electric' or 'Natural Gas' or ''Propane' or 'Fuel Oil"
Beds – Number of bedrooms
baths –Number of bathrooms
Returns:
None
Throws:
None

BAWaterHeatingBurnerCapacity

Single **BAWaterHeatingBurnerCapacity**(Curr_Fuel_Type As String, Beds As Integer, baths As Single)

This method calculates the burner capacity

Parameters:
Curr_Fuel_Type – 'Electric' or 'Gas'
Beds - Number of bedrooms
baths – number of bathrooms
Returns:

BAWaterHeatingBurnerCapacity in Btu

Throws:

None

BACoolingSetPoint

Single **BACoolingSetPoint** ( )

This method returns the BA cooling temperature in degrees F

Parameters:

None

Returns:

BACoolingSetPoint

Throws:

None

BAHeatingSetPoint

Single **BAHeatingSetPoint** ( )

This method

Parameters:

None

Returns:

BAHeatingsSetPoint

Throws:

None

BAMechVentCFMFlow

Single **BAMechVentCFMFlow**(CFA As Single, Beds As Integer)

This method returns the cfm of the benchmark mechanical vent system

Parameters:

CFA – conditioned floor area

Beds **–** number of bedrooms

Returns:

BANechVebtCFMFlow

Throws:

None

BAMechVentPower

Single **BAMechVentPower**(CFA As Single, Beds As Integer)

This method returns the Wattage of the benchmark mechanical vent system

Parameters:

CFA – conditioned floor area
Beds **–** number of bedrooms

Returns:

BAMechVentPower in Watts

Throws:

None

WaterHeatingGallonsPerDaybyMonth

WaterHeatingGallonsPerDaybyMonth(Beds As Integer, ambientannualtemp As String, baths As String, deliverytemp As String, supplytemp As String)

This method calculates the water use per day for each end-use and returns the annual total averaage

Parameters:

Beds – number of bedrooms
ambientannualtemp – average anuual outdoor temperature, degrees F
baths – number of bathrooms
deliverytemp – hot water delivery temperature, degrees F
supplytemp –hot water delivery temperature, degrees F

Returns: None

Throws:

None

Get_Equipment_Output

TBA_BenchmarkEquipCalcOut **Get_Equipment_Output** ( )

This method returns all of the equipment characteristics for a given prototype home.

Parameters:
None
Returns:
Record consisting of Equipment outputs
TBA_BenchmarkEquipCalcOut = record
Air_Handler_Location: WideString;
Duct_Material: WideString;
Supply65: WideString;
Supply35: WideString;
Air_Handler_kWhperCFM: Single;
Conditioned_Supply_R_Value: Single;
Conditioned_Return_R_Value: Single;
Total_Leakage_fraction: Single;
Total_Leakage_CFMHeat: Single;
Total_Leakage_CFMCool: Single;
DHWRecoveryEfficiency: Single;
DHWBurnerCapacity: Single;
MechVentCFMFlow: Single;
MechVentAnnualkWh: Single;
HeatingSetPoint: Single;
CoolingSetPoint: Single;
WaterUse: array[1..13] of Single;
ShowerandBath: array[1..13] of Single;
Sink: array[1..13] of Single;
Dishwasher: array[1..13] of Single;
Clotheswasher: array[1..13] of Single;
BADuct_Info: TBADuct_Info;
BACool_Info: TBACool_Info;
BAHeat_Info: TBAHeat_Info;
BAWater_Info: TBAWater_Info;
end;

Throws:
None

Set_AmbientMonthTemp

Set_AmbientMonthTemp(Jan As Single, Feb As Single, Mar As Single, Apr As Single, May As Single, Jun As Single, Jul As Single, Aug As Single, Sep As Single, Oct As Single, Nov As Single, Dec As Single)

This method sets the ambient month temperatures.

Parameters:

Jan – Ambient temperature of January
Feb - Ambient temperature of February
Mar - Ambient temperature of March
Apr - Ambient temperature of April
May - Ambient temperature of May
Jun - Ambient temperature of June
Jul - Ambient temperature of July
Aug - Ambient temperature of August
Sep - Ambient temperature of September
Oct - Ambient temperature of October
Nov - Ambient temperature of November
Dec - Ambient temperature of December

Returns:

None

Throws:

None

FindHighest

Single **FindHighest**

This method finds the highest month temperature.

Parameters:

None

Returns:

Highest value from the array

Throws:

None

FindLowest

Single **FindLowest**(ArraySize As Integer)

This method finds the lowest month temperature.

Parameters:

None

Returns:

Lowest value from the array

Throws:

None

BAMiscCalc

Description: The class contains items that may be used by multiple other classe.

Method Summary	
Single	MaxReal(a As Single, b As Single)
Single	MinReal(a As Single, b As Single)
Boolean	AllCapsCompare(MyString1 As String, MyString2 As String)

Method Detail

MaxReal

Single **MaxReal**(a As Single, b As Single)

This method returns the larger of two values.

Parameters:
a – any real number
b – any real number
Returns:
MaxReal
Throws:
None

MinReal

Single **MinReal**(a As Single, b As Single)

This method returns the smallest of two vlues

Parameters:
a – any real number
b – any real number
Returns:
MinReal
Throws:
None

AllCapsCompare

boolean **AllCapsCompare(**MyString1 As String, MyString2 As String)

This method makes a case insensitive comparison of two strings and returns true if thestrings match

Parameters:
MyString1 – any string
MyString2 –any string
Returns:
AllCapsCompare
Throws:
None

Appendix B. Water Heating Temperature Data

ID	Year	Day	Time	Sec	Temp (F)
104	2005	278	1649	1	109.3
104	2005	278	1649	2	109.2
104	2005	278	1649	3	109.8
104	2005	278	1649	4	110.7
104	2005	278	1649	5	110.6
104	2005	278	1649	6	109.4
104	2005	278	1649	7	110.4
104	2005	278	1649	8	110.5
104	2005	278	1649	9	110.5
104	2005	278	1649	10	110.7
104	2005	278	1649	11	107.8
104	2005	278	1649	12	111
104	2005	278	1649	13	111.1
104	2005	278	1649	14	111.3
104	2005	278	1649	15	111.3
104	2005	278	1649	16	111.5
104	2005	278	1649	17	111.5
104	2005	278	1649	18	111.7
104	2005	278	1649	19	110.7
104	2005	278	1649	20	111.6
104	2005	278	1649	21	111.6
104	2005	278	1649	22	111.7
104	2005	278	1649	23	111.6
104	2005	278	1649	24	111.7
104	2005	278	1649	25	111.7
104	2005	278	1649	26	111.7
104	2005	278	1649	27	110.7
104	2005	278	1649	28	111.7
104	2005	278	1649	29	111.6
104	2005	278	1649	30	109.7
104	2005	278	1649	31	111.6
104	2005	278	1649	32	111.7
104	2005	278	1649	33	111.7
104	2005	278	1649	34	111.7
104	2005	278	1649	35	111.7
104	2005	278	1649	36	111.8
104	2005	278	1649	37	111.8
104	2005	278	1649	38	111.9
104	2005	278	1649	39	111.8
104	2005	278	1649	40	110.2
104	2005	278	1649	41	110.8
104	2005	278	1649	42	112
104	2005	278	1649	43	111.8
104	2005	278	1649	44	112.1
104	2005	278	1649	45	112.1

104	2005	278	1649	46	112.2
104	2005	278	1649	47	112.3
104	2005	278	1649	48	112.5
104	2005	278	1649	49	112.6
104	2005	278	1649	50	112.8
104	2005	278	1649	51	112.2
104	2005	278	1649	52	113.3
104	2005	278	1649	53	113.7
104	2005	278	1649	54	114.2
104	2005	278	1649	55	114.6
104	2005	278	1649	56	115.3
104	2005	278	1649	57	116
104	2005	278	1649	58	117
104	2005	278	1649	59	116.7
104	2005	278	1649	60	118.9
104	2005	278	1650	1	119.9
104	2005	278	1650	2	121.1
104	2005	278	1650	3	120.8
104	2005	278	1650	4	123.2
104	2005	278	1650	5	124.1
104	2005	278	1650	6	125.1
104	2005	278	1650	7	126
104	2005	278	1650	8	126.7
104	2005	278	1650	9	127.2
104	2005	278	1650	10	127.9
104	2005	278	1650	11	128.2
104	2005	278	1650	12	128.6
104	2005	278	1650	13	128.8
104	2005	278	1650	14	129.1
104	2005	278	1650	15	128.2
104	2005	278	1650	16	129.3
104	2005	278	1650	17	129.3
104	2005	278	1650	18	129.5
104	2005	278	1650	19	129.5
104	2005	278	1650	20	129.6
104	2005	278	1650	21	129.6
104	2005	278	1650	22	129.6
104	2005	278	1650	23	131.2
104	2005	278	1650	24	129.7
104	2005	278	1650	25	129.1
104	2005	278	1650	26	128.3
104	2005	278	1650	27	129.8
104	2005	278	1650	28	129.7
104	2005	278	1650	29	129.6
104	2005	278	1650	30	129.8
104	2005	278	1650	31	131.4
104	2005	278	1650	32	130
104	2005	278	1650	33	130
104	2005	278	1650	34	130.2

104	2005	278	1650	35	130.2
104	2005	278	1650	36	130.3
104	2005	278	1650	37	130.3
104	2005	278	1650	38	130.4
104	2005	278	1650	39	130.7
104	2005	278	1650	40	130.6
104	2005	278	1650	41	130.6
104	2005	278	1650	42	130.7
104	2005	278	1650	43	130.7
104	2005	278	1650	44	130.8
104	2005	278	1650	45	130.8
104	2005	278	1650	46	130.9
104	2005	278	1650	47	128.3
104	2005	278	1650	48	131
104	2005	278	1650	49	131
104	2005	278	1650	50	129.8
104	2005	278	1650	51	128.9
104	2005	278	1650	52	131.2
104	2005	278	1650	53	131
104	2005	278	1650	54	131.1
104	2005	278	1650	55	129.6
104	2005	278	1650	56	131.1
104	2005	278	1650	57	131.1
104	2005	278	1650	58	131.1
104	2005	278	1650	59	131
104	2005	278	1650	60	131.1
104	2005	278	1651	1	131
104	2005	278	1651	2	131.2
104	2005	278	1651	3	130.9
104	2005	278	1651	4	131.1
104	2005	278	1651	5	131
104	2005	278	1651	6	131.1
104	2005	278	1651	7	131
104	2005	278	1651	8	131.1
104	2005	278	1651	9	131
104	2005	278	1651	10	131.1
104	2005	278	1651	11	131.4
104	2005	278	1651	12	131.1
104	2005	278	1651	13	131.1
104	2005	278	1651	14	129.2
104	2005	278	1651	15	131.1
104	2005	278	1651	16	131.1
104	2005	278	1651	17	131
104	2005	278	1651	18	131
104	2005	278	1651	19	129.1
104	2005	278	1651	20	131
104	2005	278	1651	21	130.9
104	2005	278	1651	22	131
104	2005	278	1651	23	130.8

104	2005	278	1651	24	130.9
104	2005	278	1651	25	130.8
104	2005	278	1651	26	130.8
104	2005	278	1651	27	129.5
104	2005	278	1651	28	130.8
104	2005	278	1651	29	130.6
104	2005	278	1651	30	130.8
104	2005	278	1651	31	130.7
104	2005	278	1651	32	130.8
104	2005	278	1651	33	130.6
104	2005	278	1651	34	130.7
104	2005	278	1651	35	130.2
104	2005	278	1651	36	130.8
104	2005	278	1651	37	130.5
104	2005	278	1651	38	128.8
104	2005	278	1651	39	130.8
104	2005	278	1651	40	130.9
104	2005	278	1651	41	130.8
104	2005	278	1651	42	130.9
104	2005	278	1651	43	129.3
104	2005	278	1651	44	131.1
104	2005	278	1651	45	131
104	2005	278	1651	46	131.2
104	2005	278	1651	47	131.2
104	2005	278	1651	48	131.3
104	2005	278	1651	49	131.3
104	2005	278	1651	50	131.4
104	2005	278	1651	51	130.2
104	2005	278	1651	52	131.4
104	2005	278	1651	53	131.4
104	2005	278	1651	54	131.4
104	2005	278	1651	55	131.5
104	2005	278	1651	56	131.5
104	2005	278	1651	57	131.5
104	2005	278	1651	58	131.4
104	2005	278	1651	59	131.5
104	2005	278	1651	60	130.4