SIMULATION OF ENERGY MANAGEMENT SYSTEMS IN ENERGYPLUS[1]

Peter G. Ellis[1], Paul A. Torcellini[1], and Drury B. Crawley[2]

[1]National Renewable Energy Laboratory, Golden, CO, USA
[2]United States Department of Energy, Washington, DC, USA

ABSTRACT

An energy management system (EMS) is a dedicated computer that can be programmed to control all of a building's energy-related systems, including heating, cooling, ventilation, hot water, interior lighting, exterior lighting, on-site power generation, and mechanized systems for shading devices, window actuators, and double facade elements. Recently a new module for simulating an EMS was added to the EnergyPlus whole-building energy simulation program. An essential part of the EMS module is the EnergyPlus Runtime Language (ERL), which is a simple programming language that is used to specify the EMS control algorithms. The new EMS controls and the flexibility of ERL allow EnergyPlus to simulate many novel control strategies that are not possible with the previous generation of building energy simulation programs. This paper surveys the standard controls in EnergyPlus, presents the new EMS features, describes the implementation of the module, and explores some of the possible applications for the new EMS capabilities in EnergyPlus.

KEYWORDS

whole-building simulation, EnergyPlus, controls, energy management system

INTRODUCTION

For the past 30 years most building energy simulation programs have shared a similar architecture: an input file describes the building form, fabric, and heating, ventilation, and air-conditioning (HVAC) systems; weather data and user schedules apply external and internal environmental forces that drive the simulation forward in time; control algorithms respond to changing conditions to meet operational requirements; and ultimately, the simulation program predicts the building performance.

Historically, control algorithms in building energy simulation programs have been designed to model typical operational strategies. Examples are thermostatic control, flow control, daily and seasonal schedules, outdoor temperature reset, economizer control, equipment load staging, and daylighting control. Most simulation input for components has predefined control sequences with only schedules and set points accessible to the user. Interactions between components are limited and are usually hard-coded in the simulation program.

With the advent of the energy management system (EMS), today's control algorithms quickly transcend the typical operational strategies that are found in most simulation programs.

An EMS is a dedicated computer that can be programmed to control all of a building's energy-related systems, including heating, cooling, ventilation, hot water, interior lighting, exterior lighting, on-site power generation, and mechanized systems for shading devices, window actuators, and double facade elements. The close cousins of the EMS, building management systems (BMS) and building automation systems (BAS), can serve the same energy functions as an EMS, but also control nonenergy systems such as for building security and life safety.

An EMS works by polling a set of sensors that retrieve data about external environmental conditions, internal building conditions, HVAC system conditions, and other equipment conditions. The sensor data become input variables for the EMS control algorithms. These algorithms are specified by using a simple programming language based on IF-THEN-ELSE statements and other logic structures (or the equivalent, if the EMS is equipped with a graphical user interface). After the EMS passes judgment, remote actuators make changes to the system operation; for example, they turn equipment on or off, change thermostat set points, and open and close valves and dampers.

In 1999, 10% of commercial buildings had an EMS. This represents about one-third of commercial building floor area (Brambly et al. 2005). An EMS is often recognized as a way to implement energy-saving control strategies and improve overall building energy performance. Average annual energy savings from an EMS are estimated at 5%-15% (Brambly et al. 2005). One common energy

cost saving application for an EMS is for demand management and load shedding.

As the demand for low-energy buildings increases, building energy simulation programs must evolve to meet the needs of design engineers. Recently a new module for simulating an EMS was added to the EnergyPlus whole-building energy simulation program (Crawley et al. 2004). An essential part of the EMS module is the EnergyPlus Runtime Language (ERL), which is a simplified programming language that is used to define the EMS control algorithms.

This paper surveys the standard controls in EnergyPlus, presents the new EMS features, describes the implementation of the module, and explores some of the possible applications for the new EMS capabilities in EnergyPlus.

STANDARD CONTROLS

Examining the inventory of standard controls in EnergyPlus is useful as background for understanding the new EMS controls. The standard controls include both low-level and high-level controls. The most complex controls in EnergyPlus are for HVAC systems.

HVAC systems are defined as a set of discrete components connected by a network of nodes. This approach offers great flexibility for connecting many system configurations.

Low-level controls

Low-level controls are meant to simulate certain closed-loop hardware controls that have a very specific job to perform. Low-level controls are often embedded in the input for an EnergyPlus object.

All EnergyPlus input data are specified by using *objects*. Each object is described in the text-based input file with an object name followed by a list of fields, delimited by commas and terminated with a semicolon. For example, the ELECTRIC EQUIPMENT object:

```
ELECTRIC EQUIPMENT,
  Zone 1 Equipment,  ! Name
  Zone 1,            ! Zone Name
  Office Schedule,   ! Schedule Name
  700,               ! Design Level {W}
  0,                 ! Fraction Latent
  0.3,               ! Fraction Radiant
  0;                 ! Fraction Lost
```

Figure 1 An EnergyPlus input object

Note that exclamation marks are used to delineate user comments which are ignored by the program.

The ELECTRIC EQUIPMENT object, for instance, has a low-level embedded control in the form of a schedule that is supplied by the user. The schedule is used to set the power level of the equipment throughout the simulation.

Many HVAC component objects also have embedded local control. Water heater objects, for instance, have a set point temperature schedule and deadband that controls how the heating element operates. Other examples are the variable air volume terminal unit objects, heat pump unitary system objects, and hydronic radiant system objects. Sometimes the embedded controls require many input fields and can be quite complex.

Several EnergyPlus objects only control other objects. These low-level objects provide some of the fundamental *system* control capabilities in EnergyPlus. Examples include the ZONE CONTROL:THERMOSTATIC object, which provides basic thermostatic control for zone heating and cooling, and the CONTROLLER:SIMPLE object, which is used to adjust water flow through a coil to achieve a target outlet air temperature. These control objects are considered low-level because they have a very specific job to perform with specific hardware. Unlike embedded controls, these system controls are external to the controlled object.

High-level controls

High-level or supervisory controls control the operation of large parts of the system. High-level controls can also jump across system boundaries to coordinate operation of the plant system and the air system.

The standard high-level HVAC system controls are typically expressed with dedicated input objects that manage and control the operation of other component objects or portions of the entire system "loop." The major types of high-level controls are:

- SET POINT MANAGERs
- SYSTEM AVAILABILITY MANAGERs
- PLANT OPERATION SCHEMEs
- DEMAND MANAGERs

Figure 2 shows a complete inventory of high-level EnergyPlus control objects.

SET POINT MANAGER objects can access data from any node in the HVAC system and use this information to calculate a set point temperature for another node in the system. The SET POINT MANAGER object is used in conjunction with a CONTROLLER:SIMPLE object.

SYSTEM AVAILABILITY MANAGER objects can also access data from any node in the HVAC system. These managers use the data to decide whether to turn an entire system loop on or off. The SYSTEM AVAILABILITY MANAGER:NIGHT CYCLE object, for example, can monitor zone temperatures

and cycle the air system to turn on if conditions become too hot or too cold.

PLANT OPERATION SCHEME objects are used to stage the plant equipment by priority according to the heating or cooling load. These objects have access to the plant loop load data and can set the availability of plant components such as chillers, boilers, and cooling towers.

DEMAND MANAGER objects are yet another type of high-level control. These are used for demand management applications that attempt to keep the total building electricity demand below a certain limit. This is accomplished by shutting off or reducing power to nonessential loads to reduce the overall demand.

All high-level control objects are simulated once per system time step. Control decisions are made based on conditions in the previous time step, i.e., lagged control. DEMAND MANAGERs, however, are the exception. Although the duration of the time step can be as short as 10 minutes, the demand limit might be exceeded within a single time step. Lagged control cannot reduce the load in time. Real demand management systems can react much more quickly to demand spikes. For this reason the DEMAND MANAGER objects in EnergyPlus use instantaneous instead of lagged control. The EMS module uses a similar type of instantaneous control and is described in more detail below.

The standard controls can accommodate a wide range of system control configurations and are useful for many common applications. Yet they do have some weaknesses. Although the controls in EnergyPlus are more flexible than those of its predecessors, BLAST and DOE-2, the control options are still rather rigid. The EnergyPlus developers cannot think of every possible control strategy. Each distinctly new control strategy requires new code development. Another weakness is that the standard control objects can be difficult to learn and use. Each object has a unique list of input fields.

All standard control objects are described in detail in the *EnergyPlus Input Output Reference* and the *EnergyPlus Engineering Reference* (UIUC, LBNL 2007a, 2007b).

EMS CONTROLS

In the hierarchy of EnergyPlus controls, the EMS module is a generalized, high-level control for all building systems. In fact, the EMS can replicate the functionality of many of the standard high-level control objects in EnergyPlus. Similar to a SYSTEM AVAILABILITY MANAGER, the EMS can turn pumps and fans on and off. Similar to a DEMAND MANAGER, the EMS can turn lights and electric equipment on and off and change the set points on zone thermostats. Similar to a SET POINT MANAGER, the EMS can change the set points on system nodes. Similar to a PLANT OPERATION SCHEME, the EMS can turn supply-side heating, cooling, and heat rejection equipment such as boilers, chillers, and cooling towers on and off.

Although the EMS can replicate the functionality of many high-level control objects in EnergyPlus, it cannot replace the low-level control objects such as CONTROLLER:SIMPLE and ZONE CONTROL:THERMOSTATIC. As is true for a real EMS, localized, low-level controls are more efficient for certain dedicated tasks such as proportional-integral-derivative (PID) control. Instead of replacing the low-level controls, the EMS can interact with control objects by changing the set points that the controllers are attempting to meet.

For high-level control, the standard and EMS controls can work together to control the same system. In some cases the standard controls may be simpler and require less input than the equivalent EMS controls. In general, the EMS controls will be used selectively when more flexibility is needed or when a certain control strategy cannot be modeled with the standard controls.

The EMS input objects are divided into two categories: hardware objects and software objects. The hardware objects describe the physical components of the EMS such as sensors and actuators. The software objects describe the logical components of the EMS such as programs and subroutines. A list of all new EMS objects is shown in Figure 3. The hardware and software objects communicate via user-defined EMS variables and ERL.

Hardware objects

The EMS hardware objects set up the physical components of the EMS. The ENERGY MANAGEMENT SYSTEM:SENSOR and ENERGY MANAGEMENT SYSTEM:ACTUATOR objects define the EMS variables that can be accessed by ERL.

The SENSOR object maps any EnergyPlus report variables or meters to an EMS variable. Report variables provide access to the complete set of exterior environmental conditions, internal zone conditions, HVAC system conditions, and all other equipment conditions. Meters provide access to utility demand and consumption data.

The SENSOR object also provides a mechanism for programming fixed schedules into the EMS. A schedule is defined by using any of the standard EnergyPlus SCHEDULE objects. The value of the schedule is then mapped to an EMS variable by using the Schedule Value report variable. When used with

the SCHEDULE:FILE:COMMA object, the SENSOR object allows data to be imported from an external file and used by the EMS. One application might be to use real experimental data to test a proposed EMS algorithm.

The ACTUATOR object maps control variables to an EMS variable. Control variables are a new feature of the EMS module that allows control signals to be sent to an EnergyPlus object. When the EMS variable corresponding to a control variable is set to a new value in an ERL program, the EMS signals for an actuator to turn a component on or off, change a set point, etc. The set of possible control variables depends on the object type that is to be controlled. All available control variables in the simulation are reported out to a new output file.

Software objects

The EMS software objects are designed to mimic the programming language used with a real EMS, yet are constrained to work within the paradigms for EnergyPlus input objects. The ENERGY MANAGEMENT SYSTEM:PROGRAM object, the ENERGY MANAGEMENT SYSTEM:INITIALIZE object, and the ENERGY MANAGEMENT SYSTEM:SUBROUTINE object are the containers for instruction blocks of ERL code. All PROGRAM objects are automatically run by EnergyPlus at every system time step. Multiple PROGRAM objects are allowed and may interact (or interfere) with each other. The PROGRAM LIST object forces a run order on the PROGRAMs.

SUBROUTINE objects can be called to run from a PROGRAM object or another SUBROUTINE object. The SUBROUTINE object is useful for encapsulating code that is called in multiple places. It is also helpful for organizing and structuring code. When the SUBROUTINE object finishes running, control is returned to the object that called it.

The ENERGY MANAGEMENT SYSTEM: GLOBAL VARIABLE object declares a global EMS variable that can be accessed by all PROGRAM objects, SUBROUTINE objects, and INITIALIZE objects. EMS variables declared by SENSOR and ACTUATOR objects are also very similar to global variables. A SUBROUTINE can emulate a function call by using global variables to store inputs and output values.

The ENERGY MANAGEMENT SYSTEM: REPORT VARIABLE object creates a custom report variable that is mapped to any EMS variable, including variables declared by a SENSOR object, ACTUATOR object, or GLOBAL VARIABLE object, or any local variable in any PROGRAM object, SUBROUTINE object, or INITIALIZE object. The custom report variable can then be reported to the output file.

The REPORT VARIABLE object is anticipated to be essential for debugging EMS algorithms. Besides its uses for controlling building systems, this object, combined with ERL, has the potential to provide a very flexible capability for custom reporting.

EnergyPlus Runtime Language

ERL is the simplified programming language that is used to define the EMS control algorithms. Every programming language comprises instructions or commands that tell the processor what to do. ERL uses a minimal set of instructions to achieve a wide range of functionality. A list of the instructions and their syntaxes is shown in Figure 4. ERL instructions are parsed and interpreted by EnergyPlus at simulation runtime.

PROGRAM, SUBROUTINE, and INITIALIZE objects use fields to store the instructions for ERL. As with most EnergyPlus objects, each field is typically given a separate line of text for the sake of readability. In this case, each field can be thought of as a separate line of code. Every field, i.e., line of code, must conform to the following rules:

- Every field contains no more than one discrete instruction
- Every instruction starts with a keyword: SET, IF, ENDIF, CALL, etc.
- All field content (keywords, variable names, etc.) is case insensitive
- A comma (or semicolon) marks the end of every instruction.

An example of a PROGRAM object with some ERL code is shown in Figure 5.

Variables are an important part of a programming language. There are five types of EMS variables: sensor, actuator, global, local, and built-in variables. All types are treated the same way by ERL and can be used interchangeably with any instruction.

All numeric variables are treated as floating point numbers. String variables will be introduced in a future release. We anticipate that this will also require other string handling instructions to be added to ERL.

A set of built-in variables provides date and time information that is not available via standard report variables, including: Year, Month, Day, Hour, Minute, Second, DayOfYear, DayOfWeek, and Julian. Predefined constant variables—True, False, On, and Off—are also available.

The counterpart of the sensor variable is the actuator variable. Sensor variables are used to get the state of building systems; actuator variables are used to set the state of building systems. When used with

actuator variables, the SET instruction performs control actions on the object to which it maps.

The rules for EMS variables are summarized as follows:

- No spaces are allowed in variable names
- Underscore _ is the only special character allowed in variable names
- Variable names are not case sensitive
- All numeric variables are treated as floating point numbers
- Sensor variables and built-in variables can be reassigned by using SET
- Actuator variables perform control actions by using SET.

All variables are dynamically typed. A variable can be of type Null, Number, String, or Array. Variable type is assigned during the SET instruction.

Expressions are key building blocks for ERL code. An expression is a sequence of variables or constants, or both, linked together by operators. An expression can always be evaluated and reduced to a single value. An expression is always the last element of a SET, IF, or ELSEIF instruction. Therefore, they can be used interchangeably for SET instructions and IF/ELSEIF instructions. That means both of the following instructions are allowed:

```
SET a = c < d
IF a - 1
```

In the case of the SET example, the value of *a* is set to 1 if *c* is less than *d*, otherwise it is set to 0. For the IF example, the IF block of instructions are executed if *a - 1* is greater than zero.

Compound expressions allow multiple operators to be sequenced or nested. For example:

```
a + b * 7 / 4.5
(a * 3 + 4) ^ 2
(a > b) && (c < d)
```

The rules for expressions are:

- An expression is a sequence of variables or constants, or both, linked by operators
- Expressions always evaluate to a single value
- Comparison operators evaluate to 1 for true or 0 for false.

IMPLEMENTATION

The EMS module is implemented with four major code components:

- EMS Manager
- Language Parser
- Runtime Interpreter
- Expression Evaluator

The core of the EMS module is the EMS Manager, which coordinates the activities of the EMS objects with the overall EnergyPlus simulation. SENSOR objects and ACTUATOR objects provide communication between the EMS Manager and the building systems.

Sensors were not difficult to implement, as they take advantage of the existing EnergyPlus report variables and meters. Pointers are used to remotely access the values of report variables in other parts of the program.

Actuators required significantly more effort to implement because there were previously no control variables for any EnergyPlus objects. A generic control variable interface for all objects was developed that can accept control messages from the EMS Manager. The interface is general enough to work with HVAC system components, as well as with non-system components such as electric lighting, electric equipment, and zone thermostats. Like sensors, the actuator interface also uses pointers, but this time it allows the EMS Manager to remotely set the control variables in other parts of the program. An initial round of actuator control variables have been implemented for a small set of EnergyPlus objects. Widespread implementation is a large task and will take place over the course of several releases.

The Language Parser is called at the beginning of the simulation to parse all instruction blocks containing ERL code. At this time the Parser issues messages about syntax errors found in the instruction blocks. Instruction blocks without errors are stored in data structures as a parsed pseudocode instruction list that is understood by the Runtime Interpreter.

During the simulation, the EMS Manager is called after the system time step is complete. At this point, all report variables and associated sensors contain updated values. The EMS Manager calls the Runtime Interpreter and passes the appropriate instruction list. Any expressions are dynamically evaluated by the Expression Evaluator by using the updated variable values for the current system time step.

Any change to an actuator variable sends a control signal back to its system component via the control variable interface. Depending on the type of component that is being controlled, the EMS Manager may force all or part of the system time step to be resimulated. Similar to the way the Demand Manager works, the EMS Manager can call for exterior equipment, the zone heat balance, or the

HVAC system to be resimulated under the new conditions. The report variables and sensors are updated and the simulation continues until completion. Under most normal control strategies the need to resimulate is relatively infrequent. However, a very poorly designed control strategy could cause the EMS Manager to resimulate nearly every time step, which would significantly increase simulation runtime.

APPLICATIONS

The new EMS capabilities have multiple applications for whole-building simulation. In general, the EMS module has applications for any advanced control scenario that requires flexibility beyond the standard controls. A major advantage of the EMS is that it can span system boundaries and allow control algorithms to integrate air system, plant system, and other objects such as electric lights and equipment. This type of coupling is not currently possible with the standard EnergyPlus controls.

One specific application that can require flexibility and system-spanning control is for demand management and load shedding. Although the standard controls do provide the DEMAND MANAGER objects, the control strategies are limited to a few fixed options. The EMS controls can provide additional flexibility.

Because the EMS module uses a simple programming language to define the control algorithms, another application is to test real-world EMS programs in simulation before they are used on a real building. Simulation, in general, provides a way to test a design concept before it is constructed. But for the EMS module there is a parallel here between input and reality that is rarely observed in building simulation. For most real-world EMS programs, whether they are defined by a programming language or a graphical user interface, the real-world control logic should be directly translatable to the ERL syntax.

At the same time that the EMS module offers a new level of flexibility, it also introduces a new level of complexity for user input. For this reason the EMS capabilities should be recognized as an advanced feature. Many common applications will continue to be served by the standard EnergyPlus controls, which are easier to configure and less susceptible to user error.

FUTURE DEVELOPMENT

Future development will include the widespread implementation of actuator control variables for more EnergyPlus objects. One important task is to implement a set of actuators that operate at the plant loop or air loop level. HVAC components on a loop can have dependencies that require the entire loop to be actuated as a whole. An interesting area for development is the addition of actuators for windows and daylighting. Window actuators could be used for natural ventilation, opening and closing windows, shades, and blinds, and setting slat positions. Daylighting actuators could dim electric lights or close window blinds to reduce solar heat gain or block glare. In addition to rudimentary actuators for turning pumps and fans on and off, actuators could be used to dynamically set the speed of variable-speed pumps and fans. Besides changing the set point on a system node, actuators could set the maximum volumetric flow rate on a node, effectively simulating a valve that can control flow through the system.

Work is also planned for extending ERL to include looping constructs and support for manipulating arrays and strings.

CONCLUSION

The new EMS module adds advanced control capabilities to EnergyPlus. The new EMS controls and the flexibility of ERL allow EnergyPlus to simulate many novel control strategies that are not possible with standard EnergyPlus control objects. The high-level scope of the EMS module allows control algorithms to span system boundaries and provide whole-building controls for a true whole-building simulation. The ability to selectively resimulate a time step allows the EMS Manager to make instantaneous control decisions, instead of having a lag of one time step.

Controls that closely mimic a programmable EMS are unprecedented for building energy simulation programs. The EMS controls add a new level of flexibility for modeling innovative control strategies that will likely become a critical part of tomorrow's low-energy buildings.

ACKNOWLEDGMENT

This work was supported by the Building Technologies Program within the Office of Energy Efficiency and Renewable Energy at the U.S. Department of Energy.

REFERENCES

Brambley, M.R., D. Hansen, P. Haves, D.R. Holmberg, S.C. McDonald, K.W. Roth, and P. Torcellini. 2005. *Advanced Sensors and Controls for Building Applications: Market Assessment and Potential R&D Pathways*, PNNL-15149, prepared for the U.S. Department of Energy by Pacific Northwest National Laboratory.

Crawley, D.B, L.K. Lawrie, C.O. Pedersen, F.C. Winkelmann, M.J. Witte, R.K. Strand, R.J.

Liesen, W.F. Buhl, Y.J. Huang, R.H. Henninger, J. Glazer, D.E. Fisher, D.B. Shirey III, B.T. Griffith, P.G. Ellis, and L. Gu. 2004. "EnergyPlus: An Update," *Proceedings of the SimBuild 2004 Conference*, August 4-6, 2004, Boulder, CO.

UIUC, LBNL. 2007a. *EnergyPlus Input Output Reference*, U.S. Department of Energy.

UIUC, LBNL. 2007b. *EnergyPlus Engineering Reference*, U.S. Department of Energy.

```
SET POINT MANAGER:SCHEDULED
SET POINT MANAGER:SCHEDULED:DUALSETPOINT
SET POINT MANAGER:OUTSIDE AIR
SET POINT MANAGER:SINGLE ZONE REHEAT
SET POINT MANAGER:SINGLE ZONE HEATING
SET POINT MANAGER:SINGLE ZONE COOLING
SET POINT MANAGER:SINGLE ZONE MIN HUM
SET POINT MANAGER:SINGLE ZONE MAX HUM
SET POINT MANAGER:MIXED AIR
SET POINT MANAGER:OUTSIDE AIR PRETREAT
SET POINT MANAGER:WARMEST
SET POINT MANAGER:COLDEST
SET POINT MANAGER:RETURN AIR BYPASS FLOW

SYSTEM AVAILABILITY MANAGER:SCHEDULED
SYSTEM AVAILABILITY MANAGER:NIGHT CYCLE
SYSTEM AVAILABILITY MANAGER:DIFFERENTIAL THERMOSTAT
SYSTEM AVAILABILITY MANAGER:HIGH TEMPERATURE TURN OFF
SYSTEM AVAILABILITY MANAGER:HIGH TEMPERATURE TURN ON
SYSTEM AVAILABILITY MANAGER:LOW TEMPERATURE TURN OFF
SYSTEM AVAILABILITY MANAGER:LOW TEMPERATURE TURN ON
SYSTEM AVAILABILITY MANAGER:NIGHT VENTILATION

UNCONTROLLED LOOP OPERATION
COOLING LOAD RANGE BASED OPERATION
HEATING LOAD RANGE BASED OPERATION
OUTDOOR DRYBULB RANGE BASED OPERATION
OUTDOOR WETBULB RANGE BASED OPERATION
OUTDOOR DEWPOINT RANGE BASED OPERATION
OUTDOOR RHPERCENT RANGE BASED OPERATION
OUTDOOR DRYBULB TEMPERATURE DIFFERENCE BASED OPERATION
OUTDOOR WETBULB TEMPERATURE DIFFERENCE BASED OPERATION
OUTDOOR DEWPOINT TEMPERATURE DIFFERENCE BASED OPERATION
COMPONENT SETPOINT BASED OPERATION

ELECTRIC LOAD CENTER:DISTRIBUTION

DEMAND MANAGER:EXTERIOR LIGHTS
DEMAND MANAGER:LIGHTS
DEMAND MANAGER:ELECTRIC EQUIPMENT
DEMAND MANAGER:THERMOSTATS
```

Figure 2 High-level EnergyPlus control objects

```
ENERGY MANAGEMENT SYSTEM:SENSOR
ENERGY MANAGEMENT SYSTEM:ACTUATOR

ENERGY MANAGEMENT SYSTEM:PROGRAM
ENERGY MANAGEMENT SYSTEM:PROGRAM LIST
ENERGY MANAGEMENT SYSTEM:SUBROUTINE
ENERGY MANAGEMENT SYSTEM:GLOBAL VARIABLE
ENERGY MANAGEMENT SYSTEM:REPORT VARIABLE
```

Figure 3 Energy Management System objects

Instruction Syntax	**Instruction Description**
`SET <var> = <expr>`	Sets the value of a variable. If <var> has not been used before, it is dynamically declared.
`IF <expr>`	Conditional decision. If <expr> evaluates to anything other than zero, the block of instructions after the `IF` are executed.
`ELSEIF <expr>`	Conditional decision that follows a regular `IF` block of instructions. If <expr> evaluates to anything other than zero, the block of instructions after the `ELSEIF` are executed.
`ELSE`	Conditional decision. Associated with an `IF` instruction, the block of instructions after the `ELSE` are executed if <expr> evaluates to zero for all preceding `IF` or `ELSEIF` instructions.
`ENDIF`	Marks the end of an `IF-ELSE` block of instructions.
`CALL <subr>`	Initiates a subroutine. Returns to the calling point when completed. Recursive calling is allowed.
`RUN <prog>`	Initiates a program. No return is expected.
`EXIT`	Prematurely exits a subroutine or program causing control to return to the caller.

Figure 4 EnergyPlus Runtime Language instruction syntax

```
ENERGY MANAGEMENT SYSTEM:PROGRAM,
  MainProgram,                      !- Name

  IF Ttank > 60,                    ! High temperature shut-off
    SET Pump = Off,
    EXIT,
  ENDIF,

  IF Tout < 0,                      ! Freeze protection control
    SET Pump = On,
    EXIT,
  ENDIF,

  SET deltaT = Tcollector - Ttank,   ! Differential thermostat
  IF deltaT > 10,
    SET Pump = On,
  ELSEIF deltaT < 2,
    SET Pump = Off,
  ENDIF;
```

Figure 5 Example of EnergyPlus Runtime Language code embedded in an object