3291

Building Emulation Computer Program for Testing of Energy Management and Control System Algorithms

William B. May, Jr.
Cheol Park

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Building Technology
Building Equipment Division
Gaithersburg, MD 20899

December 1985

Sponsored by:
Naval Civil Engineering Laboratory
U.S. Department of Defense

ce of Building and Community Systems
. Department of Energy

NBSIR 85-3291

BUILDING EMULATION COMPUTER PROGRAM FOR TESTING OF ENERGY MANAGEMENT AND CONTROL SYSTEM ALGORITHMS

William B. May, Jr.
Cheol Park

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Building Technology
Building Equipment Division
Gaithersburg, MD 20899

December 1985

Sponsored by:
Naval Civil Engineering Laboratory
U.S. Department of Defense

Office of Building and Community Systems
U.S. Department of Energy

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

ABSTRACT

A building emulator can be used to test energy management and control systems (EMCS). The emulator uses a computer program to simulate the responses of a building including the equipment, building space, and building envelope to EMCS commands. Building model software for the emulator has been developed at the National Bureau of Standards (NBS) in an effort to assist the United States Naval Civil Engineering Laboratory (NCEL), which is developing a sophisticated building emulator.

The concept of the building emulator and the building emulator computer program are described in this report. The program includes the weather, the air handling unit, the zone, and the comfort model. In addition, the energy compilation routine is also included. The models presented here are simplified models. With these abridged models, a single zone building with exterior walls and a single deck air handling unit are simulated. A complete FORTRAN source code of the building emulator computer program is appended.

KEY WORDS: air handling unit simulation model; building emulator; building space zone model; computer simulation; EMCS algorithms; energy management and control systems (EMCS); HVAC emulator; local equipment simulation model; testing EMCS algorithms.

ACKNOWLEDGEMENT

The authors are indebted to the Naval Civil Engineering Laboratory, Department of Defense, and the Building Systems Division, Department of Energy, for funding this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CONVERSION FACTORS FROM METRIC (SI) TO ENGLISH UNITS

Physical Characteristic	To Convert from	To	Multiply by
Length	m	ft	3.28
Area	m^2	ft^2	10.76
Velocity	m/s	mph	2.24
Temperature	°C	°F	$t_F = 1.8t_C + 32$
Temperature difference	°C	°F	1.8
Energy	J	Btu	0.948×10^{-3}
Power	W	Btu/hr	3.41
U-value	$W/m^2 \cdot °C$	$Btu/hr \cdot ft^2 \cdot °F$	0.176
Thermal resistance	$m^2 \cdot °C/W$	$hr \cdot ft^2 \cdot °F/Btu$	5.678
Pressure	kPa	in. Hg	0.296

1. INTRODUCTION

Advances in computer and communications technology have made feasible the use of computers for controlling the heating, ventilation, and air conditioning (HVAC) systems of buildings. Such control systems are usually called energy management and control systems (EMCS). As for any building equipment, it is desirable to have standard testing and rating methods for EMCS. One benefit of such standards is that a more intelligent choice of an EMCS for a building can be made if systems from several manufacturers are tested using standard methods. Once an EMCS is installed, acceptance testing of the system is possible by comparing field test results to design specifications.

1.1 The Concept of a Building Emulator

One approach to testing of EMCS systems is the use of a building emulator. A building emulator is a special computer connected to the sensor inputs and command outputs of an EMCS. No actual HVAC equipment or building is connected to the EMCS. The emulator uses a computer program to simulate the responses of a building (including the equipment, building space, and building envelope) to EMCS commands. The EMCS then controls this simulated building as if it were an actual one. The advantages of using an emulator are that an EMCS may be tested with any type of building if a model is available, and tests can be repeated on different EMCS under the same conditions. Although experience with emulators is limited, it appears that the emulator testing approach is the most desirable in terms of the cost of testing, reproducibility of results, and control of test conditions.

A building emulator can be used to test EMCS algorithms, which may be defined as the software programs in the EMCS which use the EMCS sensors to collect information, make decisions based on current and past information and parameters, and make changes to the building through the EMCS actuators. EMCS algorithms can either be direct digital control (DDC) to control valves and dampers directly, for example, or supervisory to determine start/stop times and setpoints, for example. If a building emulator is used to test EMCS algorithms, it is not necessary to know the exact structure of the algorithm software. Since EMCS software is often proprietary, this is a definite advantage.

Before a building emulator can be used, it must be possible to connect an emulator to an EMCS to be tested. The recommended approach is the connection of the emulator to the EMCS in place of the regular EMCS sensors and actuators. Since it is likely that the EMCS sensors are electrical in nature, they can be replaced with voltage or current sources under the control of the emulator. The EMCS will be unable to distinguish between an actual sensor producing an electrical signal and the emulator producing the same electrical signal. This method allows the EMCS algorithm to be tested directly without the effects of sensors. This implies that a building emulator cannot be used to test the instrumentation on an EMCS since no sensors or actuators are involved.

1.2 Existing and Planned Building Emulators

Two building emulator prototypes have been built in order to demonstrate the concept of EMCS testing using an emulator. At the United States National Bureau of Standards (NBS), a simple building emulator was constructed in 1983. This emulator demonstrated the basic concept of a building emulator but was not transportable, was limited in capacity, and had a model which was too simple for quantitative algorithm testing.

A second example of a prototype emulator was developed by a contractor for the United States Naval Civil Engineering Laboratory (NCEL) in 1983 [Wise (1984)]. This device, which was referred to as an EMCS exerciser, was designed to be transportable and was used in field tests of actual EMCS installations. Complex software in the exerciser allowed an operator to easily prepare for and run tests. The tests were intended to determine if EMCS supervisory algorithms were operating properly. The exerciser produced reports of the time of occurrence and nature of all EMCS actions during the test. These reports were compared to reports from the EMCS to determine if any problems existed. The unit was not designed to allow a quantitative testing of algorithms, only to determine if the algorithms were operating and controlling the correct command outputs. The exerciser did work as designed and was capable of revealing areas requiring attention in real EMCS installations.

NCEL, with the assistance of NBS, is currently developing a building emulator of greater sophistication than the examples given above. This emulator will have the field transport and general connection features of the NCEL exerciser plus a detailed building simulation model and improved control and evaluation software. NBS is responsible for developing the building model to be used in the NCEL emulator. This report is intended to describe the building model software which has been developed by NBS. The NBS building model does not simulate a complete building, but simulates a single building zone with exterior walls and an air handling unit.

The NCEL/NBS emulator will be used for either factory or field acceptance tests of EMCS algorithms. A factory test will be performed at an EMCS manufacturer facility to determine the capabilities of algorithms on a specific EMCS. A field acceptance test will be used to determine whether an EMCS installed at a specific site is operating in accordance with design specifications. The initial version of the emulator will be designed to test supervisory EMCS algorithms and will not test DDC algorithms.

1.3 Testing of EMCS Algorithms with the NCEL/NBS Building Emulator

Eight types of supervisory control algorithms have been selected for testing with the NCEL/NBS building emulator. The concepts of the algorithms will not be discussed in this report. Discussion of these EMCS algorithms may be found in other reports [May (1983), Park (1983), Park, Kelly, Kao (1984), Park (1984), and May (1984)]. The types of algorithms to be tested are listed below:

1. Scheduled Start/Stop
2. Duty Cycling
3. Demand Limiting
4. Optimum Start/Stop
5. Day/night Setback
6. Economizer
7. Ventilation/Recirculation
8. Air Setpoint Reset

The emulator model described in this report will not currently support the testing of the demand limiting algorithm. A model of electrical demand in a building would have to be added to the emulator simulation software.

The test procedure to be used for testing EMCS algorithms will determine the exact characteristics that the NCEL building emulator will have. The final form of the test procedure is not known as of the writing of this report. The test procedure cannot be determined entirely by objective means and must result from negotiations between building owners, equipment manufacturers, testing personnel, and standards organizations. For the NCEL emulator it will be assumed that the test procedure will be based on the concept of testing of computers using a 'benchmark' computer program which exercises various operations of any computer on which it is run. Like a program which runs at different speeds depending on the computer on which it is executing, an EMCS algorithm will behave differently depending on the building and climate where it is tested. If 'benchmark' or 'reference' buildings are used for a building emulator, then results of testing different algorithms of the same type on these buildings may be compared.

The building simulation program in the emulator will simulate the response of a building to EMCS control actions. The emulator will be used to provide reproducible predictions of energy use and temperature levels for a set of hypothetical 'reference' buildings which are being controlled by the EMCS system under test.

1.4 Software Components of the NCEL/NBS Building Emulator

The NCEL/NBS emulator computer will employ a multi-tasking disc operating system which will allow several software tasks or processes to effectively appear to be running concurrently. When the emulator is being used to test an EMCS algorithm, there will be two important processes running. One of these is the building simulation program which is described in this report. The other process is an input/output (I/O) program which will read from and write to the hardware connected to the EMCS sensors and actuators. These two processes will have a common data area in computer memory that both may read from and write to.

If the EMCS sends a command by changing the state or signal level of an EMCS command output, the I/O program detects this and changes the data area that the simulation program uses for command input variables such as start/stop fan, or supply air setpoint. Based on the input variables and model parameters, the simulation program predicts values for variables such as temperatures in the building and HVAC equipment and status points. The simulation program places these values in the common memory area. The I/O program reads these values from the common area and sets the signal levels on the EMCS sensor inputs.

In addition to the model and I/O software, the NCEL emulator will have control software. An operator control program will be used to start and stop the simulation and I/O programs, select parameter files to be used, and determine duration of testing. At the end of tests the simulation program will deposit accumulated energy use by the emulated building into data files. Evaluation software will be used to examine test results and produce evaluations of specific EMCS algorithms such as scheduled start/stop, demand limiting, or setpoint reset.

The next section of this report describes the model software for the emulator in detail. No additional information will be given on the other software components of the emulator since these were not developed at the time this report was written.

2. BUILDING SIMULATION COMPUTER PROGRAM

The building emulator model computer program that was completed by NBS was designed to simulate any type of HVAC system and all possible modes of heat transfer in a zone. However, not all subroutines were completed in order to create a more simplified version. This simplified version will be termed the abridged version. The extent of simplifications will be noted in the sections following.

The building emulator building simulation computer program was written completely in the high level computer language FORTRAN 77 and is appended in the APPENDIX. This section will describe the actual computer program that was developed but will not present the specific equations used to model the building. The emphasis in this chapter will be on describing the data structures, data interface, programming logic and component parts of the program. Following chapters will present detailed equations for the component models of the simulation program.

2.1 Data Structures and Interface to Building Emulator

The building emulator model program is connected to the other programs in the emulator by three data paths. These are the common data area, parameter input files, and report output files. In terms of the building model, the parameter input files control duration of the simulation, timestep, initial conditions, and configure the building systems to be simulated. The report files are the program output, and the common data area represents the state of the simulation and allows input of time dependent boundary conditions.

The common data area is implemented in the emulator model program as a FORTRAN common block. Within the model program, this common block is given the name STATE1 and contains 100 real variables and 100 logical variables. The common block STATE1 contains the 'state vector' for the model. This means that at any time, the values of the variables in STATE1 describe the state of the simulation. Tables 2-1 through 2-4 list all of the variables in the state vector and the engineering units used. The number on the left is the position of the variable in the state vector array. Not all of the state vector positions are used to allow for any future expansion of the model. The variables in table 2-1 are all set by the emulator model program. Selected values in the common area will be converted to electrical signals on the EMCS inputs.

The variables listed in table 2-2 are energy variables which contain cumulative energy used by the building in various categories. This information is written to report files at the end of an emulator test run. More information on energy variables is given in section 7.

Table 2-3 lists variables which the emulator model program uses as input (independent variables). The values of these variables in the common area are set depending on what commands have been received from the EMCS. Some of the variables are logical and some are real variables. The logical variables have two states for use in controlling a two-state device such as a fan. The real variables are used to determine values such as setpoints.

The variables in table 2-4 are set by the emulator program but are not converted to signals for the EMCS. These are comfort variables which are written to a report file at the end of a test period.

Table 2-1. Emulator model state vector - Analog sensor variables

01: TEMPERATURE of outside air (F)
02: HUMIDITY RATIO of outside air (lbw/lbda)
03: ENTHALPY of outside air (Btu/lb)
04: Windspeed (MPH)
06: TEMPERATURE of zone air (F)
07: HUMIDITY RATIO of zone air (lbw/lbda)
08: TEMPERATURE of zone wall interior (F)
09: TEMPERATURE of Zone wall surface (F)
10: TEMPERATURE of zone glass (F)
20: TEMPERATURE of return air (F)
21: ENTHALPY of return air (BTU/lb)
22: HUMIDITY RATIO of return air (lbw/lbda)
23: TEMPERATURE of mixed air (F)
24: ENTHALPY of mixed air (BTU/lb)
25: HUMIDITY RATIO of mixed air (lbw/lbda)
26: TEMPERATURE of cooling coil discharge air (F)
27: ENTHALPY of cooling coil discharge air (BTU/lb)
28: HUMIDITY RATIO of cooling coil discharge air (lbw/lbda)
29: TEMPERATURE of heating coil discharge air (F)
30: ENTHALPY of heating coil discharge air (BTU/lb)
31: HUMIDITY RATIO of heating coil discharge air (lbw/lbda)
32: TEMPERATURE of supply air (F)
33: ENTHALPY of supply air (BTU/lb)
34: HUMIDITY RATIO of supply air (lbw/lbda)
35: ENTHALPY of return air plus ventilation air(BTU/lb)
51: MASS FLOW RATE of return air (lb/s)
52: MASS FLOW RATE of supply air (lb/s)
53: MASS FLOW RATE of ventilation air (lb/s)
60: HEATING POWER to heating coil (Btu)
61: COOLING POWER from cooling coil (Btu)
62: HEATING POWER to local zone equipment (Btu/s)
63: COOLING POWER from local zone equipment (Btu/s)

The input parameter files are read by the emulator model program during the initalization of the program. These files are described in section 2.3. The input parameters files are either model parameter files or test control parameter files.

2.2 General Program Organization and Control

The emulator building model computer program is written in the form of a FORTRAN subroutine which can be called from the main program section of the emulator model process running in a multi-tasking system (see section 1.4). The main logic diagram for the emulator model is shown in figure 2-1. It is assumed that the emulator subroutine is called repeatedly without any time delays between calls.

Table 2-2. Emulator model state variables - Energy variables

64: HEATING ENERGY (requirements) (Btu)
65: COOLING ENERGY (requirements) (Btu)
66: HEATING ENERGY (economizer) (Btu)
67: COOLING ENERGY (economizer) (Btu)
68: HEATING ENERGY (load) (Btu)
69: COOLING ENERGY (load) (Btu)
70: HEATING ENERGY (reheat) (Btu)
71: COOLING ENERGY (recool) (Btu)
72: ELECTRICAL ENERGY to fans (kWh)
73: HEATING ENERGY to local zone equipment (Btu)
74: COOLING ENERGY from local zone equipment (Btu)

Table 2-3. Emulator model state variables - Command variables

80: CONTROL POINT ADJUSTMENT supply air temperature (F)
81: CONTROL POINT ADJUSTMENT zone temperature (F)
82: CONTROL POINT ADJUSTMENT outside air damper position (F)
83: CONTROL POINT ADJUSTMENT supply air RH (%)
Logical 01: COMMAND air handling unit on/off
Logical 02: COMMAND economizer on/off
Logical 03: COMMAND ventilation on/off
Logical 04: COMMAND zone setback on/off
Logical 10: STATUS air handling unit on/off
Logical 11: STATUS zone occupied/unoccupied

Table 2-4. Emulator model state variables - Comfort variables

90: COMFORT time outside of comfort range (S)
91: COMFORT maximum dry bulb temperature (F)
92: COMFORT minimum dry bulb temperature (F)
93: COMFORT relative humidity at maximum temperature (%)
94: COMFORT relative humidity at minimum temperature (%)

The emulator program consists of three parts. These are the model equations section, the control section, and the initialization section. During the initialization of the program, a simulation time step, for example thirty seconds, is selected. In a conventional simulation, the control section of the program would advance the variables which contain the time of day and cause the model equations to be solved. The conventional simulation will execute at the full speed of the computer. For an emulator, however, the simulation is artificially slowed so that the time intervals as measured by a real time clock between solutions of the model correspond to the value of the simulation time step. The control software for an emulator must obtain the current time of day from a real-time clock in the computer and only allow the model equations to be solved when a real amount of time has passed.

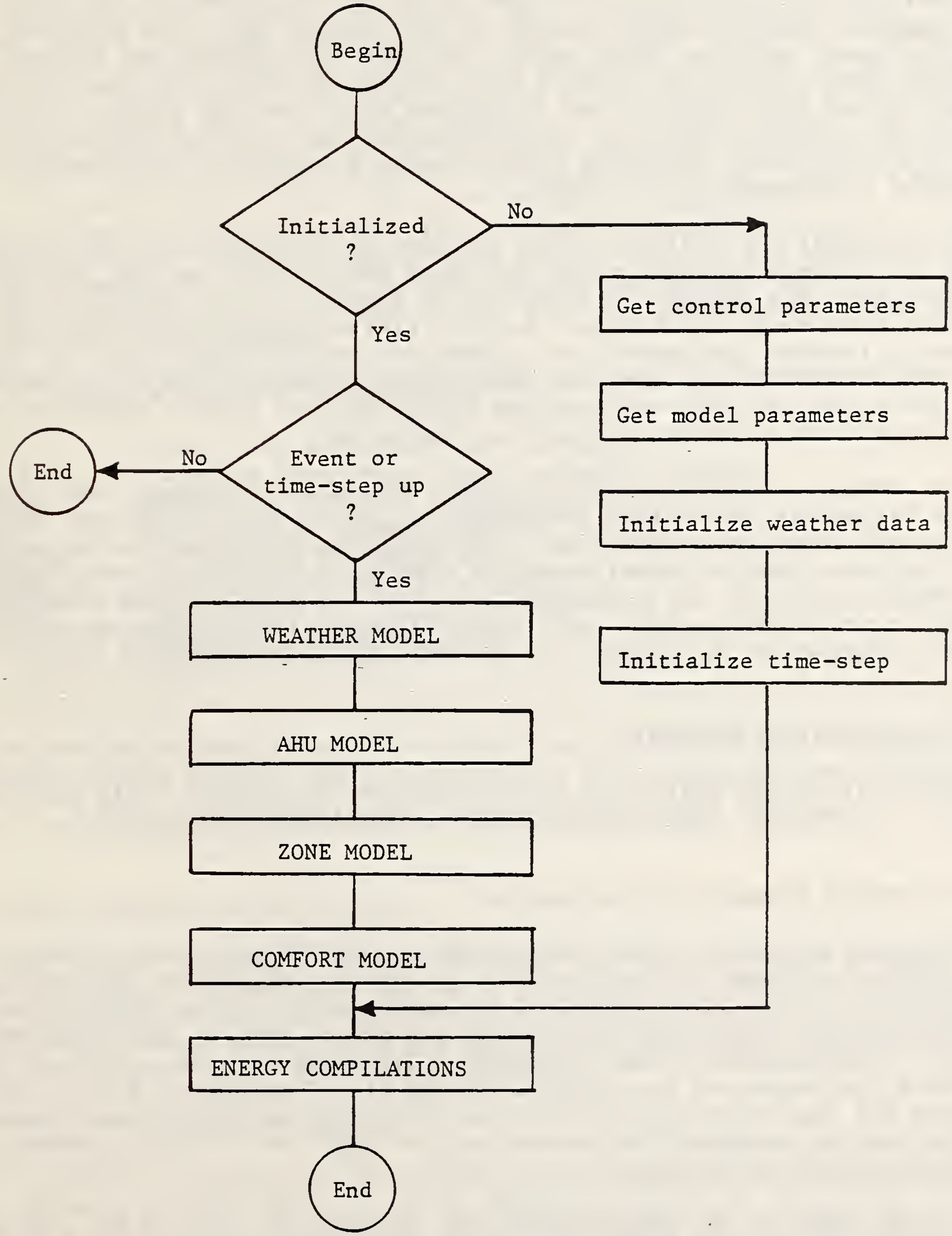

Figure 2-1. Logic diagram of the emulator model program

In addition to monitoring a real-time clock for the passage of the time-step, the emulator building model program control section must also determine if a command event has taken place before the time-step interval has elapsed. If a command event occurs, the program must temporarily adjust the time step to be equal to the time interval that has passed since the most recent solution of the model equations and cause the model equations to be solved immediately. If this is not done, there may be unacceptable delays in the response of the emulator to commands from the EMCS.

Each time that the emulator model subroutine is called, a determination is made as to whether initialization has taken place. If it has not, the program initializes. This process is described in section 2.3. If initialization has already occurred, the program control section determines if either the timestep interval has passed, or a command event has taken place. If neither of these conditions has occurred, then the subroutine is exited. If either of the conditions has occurred, then the model equations section causes the model equations to be solved for the current time of day.

After either initialization or model equations solution, the program calls the energy compilation subroutine. If called after equations solution, the energy compilation subroutine determines the energy used by the emulated building since the last model equations solution and stores a cumulative total of energy used since the beginning of the emulator model program execution. If called after initialization, the cumulative energy totals are reset. After the energy compilation if finished, the emulator model subroutine is exited.

2.3 Initialization Routines

The initialization section of the emulator model program loads simulation control parameters, model parameters, and initializes time variables.

2.3.1 Control Parameter Initialization

The control parameter input subroutine, named CONPAR, is called during the initialization phase to load control parameters. The control parameters are previously stored in a file called the configuration file. CONPAR opens the configuration file and reads the control parameters from it. Table 2-5 shows a sample configuration file. All reading from the file is performed using FORTRAN list-directed read operations and therefore quotes must be placed around the character information, but otherwise no formatting or special delimiters are required. The entries on the right are strictly comments and are not read by the program.

The first entry in the configuration file is the start time of the test using the building emulator. In the example configuration file, this is 12:30. The next entry is the duration of the test in days, hours, minutes, and seconds. In table 2-5, the example test duration is one day and three hours, or a total of 27 hours. The next four entries in the configuration file are the names of files containing the model parameters. The contents of these files are discussed in the following sections. The names of these files would normally be changed to select a different set of model parameters to allow a different building to be simulated, or the same building to be simulated in a different climate. The seventh entry in the configuration file is the name of the file used to initialize the state vector. The eighth configuration file entry is the weather 'season' to use for the simulation. This will be discussed more fully in section 2.3.2.5. The last entry in the configuration file is the simulation time-step. The time-step is discussed in section 2.2 and section 2.4. The smaller the timestep, the greater the accuracy of the results, but the program execution time rises quickly with decreasing time step. A larger timestep decreases execution time up to a point. Beyond this point the execution time does not change.

Table 2-5. Sample simulation configuration file

```
 12 30  0        'start time: hours, minutes, seconds'
1 3  0  0        'execution time: days, hours, minutes, seconds'
'HVAC.PAR'       'BUILDING HVAC SYSTEM PARAMETER FILE'
'USE.PAR'        'BUILDING USE PARAMETER FILE'
'CLIMATE.PAR'    'WEATHER DATA PARAMETER FILE'
'SHELL.PAR'      'BUILDING SHELL PARAMETER FILE'
'WINTER.INI'     'STATE VECTOR INITIALIZATION FILE'
'WINTER      '   'SEASON TO USE FOR EMULATION'
20               'TIMESTEP IN SECONDS'
```

After the control parameters have been read, the CONPAR subroutine calculates the absolute ending time for the emulator test run by adding the test duration to the start time. The ending time is used to determine when to halt the execution of the emulator model program and write the report files.

2.3.2 Model Parameter Initialization

After the control parameters are initialized, the initialization section of the emulator model program calls the subroutine named PARAMA. This subroutine is used to initialize all model parameters. The model parameters are stored in several FORTRAN common blocks which are installed in the PARAMA subroutine and in the various model equation solution subroutines.

2.3.2.1 State Vector Initialization

The first function performed by the PARAMA subroutine is the initialization of the state vector. The variables in the building emulator simulation are located in a large array in a FORTRAN common block. This state vector contains 100 real variables and 100 logical variables. During the solution of the model equations, some of the state variables are calculated completely from the current values of other variables and do not need to be initialized. Other state variables are calculated from the previous value of the state variable one time step back as well as from the current value of other variables. This latter type of variable must be given an initial value. The state vector variables are initialized using values read from the state vector initialization file. Table 2-6 shows a sample initialization file.

Table 2-6. Sample state vector initialization file

06	52.0	'TEMPERATURE of zone air (F)'
07	0.008	'HUMIDITY RATIO of zone air (lbw/lbda)'
08	21.0	'TEMPERATURE of zone wall interior (F)'
09	48.0	'TEMPERATURE of Zone wall surface (F)'
10	73.0	'TEMPERATURE of zone glass (F)'
80	81.0	'CONTROL POINT ADJUSTMENT supply air temperature (F)'
81	55.0	'CONTROL POINT ADJUSTMENT zone temperature (F)'
83	50.0	'CONTROL POINT ADJUSTMENT supply air RH (%)
92	200.0	'COMFORT minimum dry bulb temperature (F)'
01	.FALSE.	'COMMAND air handling unit on/off'
02	.FALSE.	'COMMAND economizer on/off'
03	.FALSE.	'COMMAND ventilation on/off'
04	.FALSE.	'COMMAND zone setback on/off'
10	.FALSE.	'STATUS air handling unit on/off'

There are three entries on each line of the initialization file. The first entry on the line is the position in the state vector to be initialized. This is a number from 1 to 100. It is assumed that the real numbers are initialized first. The lines in the file must have state vector position numbers in ascending numerical order. When a position number for a line is lower than the number in the preceding line, it is assumed that the following lines will now be used to initialize logical variables.

The second entry on the lines in the initialization file is the initial value to be assigned to the state vector variable. This is a real number constant for the real state variables and a logical constant for the logical state variables. The third entry on the initialization file lines is a comment identifying the name of the state variable. This is not read by the program.

2.3.2.2 Building Use Parameter Initialization

The second function performed by the PARAMA subroutine is the initialization of building use parameters. The building use parameter file contains information on the occupancy and equipment operating times for the building. An example of a building use parameter file is listed in table 2-7.

In the building use parameter file each line is used to determine a change in the usage status of the building. Between the times indicated by the entries, the usage status is assumed to remain the same. The first column indicates the time in decimal hours that the change in building usage is to occur. The second column indicates the level of lighting in the space in watts. The third column determines the type of lighting used in the building. 'F' indicates fluorescent, and 'I' indicates incandescent. The fourth column is the power used by equipment in the zone in watts. The fifth column is the ratio of radiated energy to total energy for the equipment. The sixth column is the zone moisture gain rate due to the equipment in pounds of water per hour. The last five columns are counts of the number of occupants in one of five activity levels with Activity 5 being the most active. The activity levels are described in section 5.4.5.2.

Table 2-7. Sample building use parameter file

HRS	LIGHT(W)	TYPE	EQUIP,SENS(W)		EQUIP,LAT(LB/H)	ACT 1	ACT2	ACT3	ACT4	ACT5
0.0	100.	'F'	0.	0.5	0.	0	0	0	0	0
8.0	500.	'F'	200.	0.5	0.25	16	8	4	0	0
17.0	100.	'F'	0.	0.5	0.	0	0	0	0	0

After the parameters have been read from the file. The PARAMA subroutine performs some simplifying calculations. The purpose of the calculations is to determine the sensible heat gain rate from equipment, the sensible heat gain rate from people; the moisture gain rate from equipment, and the moisture gain rate from people for each usage state. The equations used are described in section 5.4.5.2. The calculated building use parameters are stored in the common block USE in five arrays. The five arrays contain the above calculated parameters and the time of occurrence for each usage state. A parameter in the program determines the maximum number of usage states.

2.3.2.3 Building Shell Parameter Initialization

After initializing the building use parameters, the PARAMA subroutine initializes the building shell parameters. The building shell parameter file contains parameters which describe the characteristics of the zone, zone walls, and zone windows, for the building to be emulated. Parameters describing the thermal capacity and insulation value of the zone enclosure are in this file. Also in the file are parameters for air infiltration and moisture absorption. An example of the contents of the file is given in table 2-8. The meaning of the entries is assumed to be explained by the comments on each line and will not be discussed here. Only the first value on each line is read by the program. No additional calculations are performed on the building shell parameters in subroutine PARAMA. The parameters are stored in a FORTRAN common block with the name SHELL.

2.3.2.4 HVAC System Parameter Initialization

After initializing the building shell parameters, the PARAMA subroutine initializes the building HVAC system parameters. The HVAC System parameter file contains a description of the heating and cooling equipment available in the building to be emulated. An example of this parameter file is given in table 2-9. The meaning of the entries is assumed to be explained by the comments on each line. Only the first value on each line is read by the program.

Table 2-8. Sample building shell parameter file

1000.	'FT^2'	'Area of the building wall exposed to the outside'
4.05E-4	'BTU/S*F*FT^2'	'inner wall film coeff.(STILL AIR, VERTICAL) R0.68'
1.11E-3	'BTU/S*F*FT^2'	'outer wall film coeff.(7.5 MPH WIND) R0.25'
0.282	'FT'	'thickness of wall outside the mass center'
0.695	'FT'	'thickness of wall inside the mass center'
3.00E-2	'BTU/S*F*FT'	'wall conductivity > the mass center (BRICK)
7.12E-5	'BTU/S*F*FT'	'wall conductivity < the mass center (MASONRY)'
100.0	'BTU/F'	'thermal capacity of zone furnishings (300LB WOOD)'
25000.	'FT^3'	'volume of air in zone (50x50x10 ft) '
0.19	'BTU/LB*F'	'specfic heat of wall outside the mass center'
130.0	'LBM/FT^3'	'density of wall outside the mass center'
0.22	'BTU/LB*F'	'specfic heat of wall inside the mass center'
52.8	'LBM/FT^3'	'density of wall inside the mass center'
0.5	'ACH'	'base air infiltration rate without wind and thermal'
3.0		'furnishings moisture absorption mass relative to air'

Table 2-9. Sample HVAC system parameter file

68.75	'ft^3/sec.'	'Air handling unit fans rated supply air volume'
15.0	'kW'	'power requirement of supply fan at rated volume'
10.0	'kW'	'power requirement of return fan at rated volume'
'REHEAT'		'Local equipment type'
05.0	'Btu/sec.'	'Local equipment capacity, heating'
0.40000	'Btu/sec. F'	'Local equipment controller proportional gain'
0.00010	'Btu/s2 F'	'Local equipment controller integral gain'
60.0	'sec.'	'Local equipment controller sensor time constant'
.FALSE.		'Variable Air Volume System?'
.TRUE.		'Return Air Fan in system?'
0.05	'Btu/sec.'	'Return Air Fan air heating rate'
0.10		'Minimum ventilation air, fraction of supply air mass'
.FALSE.		'Dual Deck System?'
0.06	'Btu/sec.'	'Supply Air Fan air heating rate'
.FALSE.		'Humidity Control?'
10.0	'Btu/sec.'	'Cold Deck Heating Coil Capacity'
15.0	'Btu/sec.'	'Cold Deck Cooling Coil Capacity'
20.	'F'	'Local equipment controller setpoint setback'

2.3.2.5 Climate Parameter Initialization

After initializing the building HVAC system parameters, the PARAMA subroutine initializes the building climate parameters. The building climate parameter file contains the parameters used to simulate the weather to which the emulated building is exposed. The information in the file is for a representative day at five different times of the year for a particular climatic area. These times of the year are termed 'seasons' and correspond to winter, late winter, spring, early summer, and summer. During any particular test, parameters for one season, which is selected by an entry in the emulator model configuration file, will be used to generate a one day variation in weather conditions. This day of weather will be repeated for any other days that the test is run. An example climate file is given in table 2-10. The first line in the file identifies the climate description. This is followed by the five groupings of parameters for five seasons. The season title is the first season information entry. The other entries are explained by the adjacent comments. Additional information about these parameters is given in chapter 3.

2.4 Simulation Control Routines

The simulation control section of the program is found in a subroutine given the name ECTROL. ECTROL has two output arguments, TIMEST and GO. TIMEST is an integer variable which returns the current time step for the simulation to use. GO is a logical variable which is returned as true if the model equations should be solved and false if the equations should not yet be solved.

ECTROL uses two subroutines, TIMINC and NUSTEP, to determine values for variables TIMEST and GO. TIMINC has two logical output arguments, STEPDN and TIMEUP. If a time interval equal to the simulation time step has elapsed since the last solution of the model equations, STEPDN is set true, otherwise it is set false. If the duration of the emulator test specified in the emulator configuration file has elapsed, TIMEUP is set true, otherwise it is set false. If TIMEUP is true, ECTROL calls REPORT, which generates the emulator model report files. EVENT is a global logical variable which is true if the command inputs in the state vector have changed since the last model equations solution, or false otherwise. If either EVENT or STEPDN is true, GO is set to TRUE and ECTROL calls NUSTEP. NUSTEP returns in the variable TIMEST the time elapsed since the last solution of the model equations.

2.5 Building Component Model Routines

The solution of the model equations is effected by successive calls to WEATHR, the weather simulation routine, AHU, the air handling unit simulation routine, ZONE, the zone simulation routine, COMFRT, the comfort model routine, and CMPILE, the energy use compilation routine. All of the models, with the exception of ZONE, are steady state models not involving differential equations.

Table 2-10. Sample weather data parameter file

```
'Cold humid winter, hot humid summer, moderate sun'

'WINTER'
35.0      'Maximum temperature, F'
15.0      'Minimum temperature, F'
14.0      'Time of maximum temperature, hrs'
0.003     'Humidity ratio
500.      'Peak solar heat gain on horizontal'
10.0      'Average windspeed, MPH'

'LATE WINTER'
40.0      'Maximum temperature, F'
20.0      'Minimum temperature, F'
14.0      'Time of maximum temperature, hrs'
0.003     'Humidity ratio
1000.     'Peak solar heat gain on horizontal'
10.0      'Average windspeed, MPH'

'SPRING'
60.0      'Maximum temperature, F'
35.0      'Minimum temperature, F'
13.0      'Time of maximum temperature, hrs'
0.004     'Humidity ratio
1700.     'Peak solar heat gain on horizontal'
11.0      'Average windspeed, MPH'

'EARLY SUMMER'
80.0      'Maximum temperature, F'
60.0      'Minimum temperature, F'
14.0      'Time of maximum temperature, hrs'
0.010     'Humidity ratio
2000.     'Peak solar heat gain on horizontal'
 9.0      'Average windspeed, MPH'

'SUMMER'
85.0      'Maximum temperature, F'
60.0      'Minimum temperature, F'
14.0      'Time of maximum temperature, hrs'
0.013     'Humidity ratio
1800.     'Peak solar heat gain on horizontal'
 8.0      'Average windspeed, MPH'
```

WEATHR determines the outside air temperature using a sinusoidal approximation. All other values such as humidity ratio and windspeed are assumed constant. Solar gain is calculated geometrically. Section 3. describes the equations solved in the weather model.

AHU is used to determine the energy required to heat or cool a mix of return and outdoor air to the supply air setpoint. The model has heating coils, cooling coils, and an outside air damper. These devices are assumed to be under perfect sequenced control. There is no local economizer control of the dampers. Section 4 describes the equations for the air handling unit model.

The ZONE subroutine performs a solution of 5 simultaneous equations of which three are differential equations and two are steady-state. These equations describe the dynamics of heat and moisture transfer between perfectly mixed zone air and a two-node wall and the outside environment. A model of local zone heating or cooling equipment under PI control is included in subroutine ZONE. Details of the solution technique and the equations used in subroutine ZONE are discussed in section 5.

The routine COMFRT compares the current zone space conditions to the ASHRAE comfort zone and determines if the zone is comfortable. The comfort model equations are described in section 6.

The subroutine CMPILE, using values from the state vector, accumulates the energy used by the building in various categories. The equations used to calculate energy are described in section 7.

3. WEATHER MODEL

3.1 Assumptions and Purpose of Weather Model

The purpose of the weather model is to provide values for the dry bulb temperature, humidity ratio, windspeed, and solar radiation on a vertical surface for each timestep in the simulation. It is assumed that tests of an EMCS system will require the generation of one to five days of continuous weather data. For such a time period, all of the days in the period will be assumed to have identical weather. The humidity ratio and windspeed will be assumed to remain constant throughout the test period. The dry bulb temperature is assumed to vary smoothly in a sinusoidal manner with a period of one day. Solar radiation is calculated from geometric relationships and statistical values for direct solar radiation.

Model parameters for the weather model were listed in table 2-10. These parameters allow the weather model to be modified to represent different climates and seasons within climates to which a building might be exposed within the constraints listed above. The parameters for humidity ratio and windspeed specify the constant value that these variables will have throughout the day. Other specific parameters will be discussed in the appropriate sections below.

3.2 Outside Air Temperature

The model parameters which determine the variation of outside dry bulb temperature with time for the weather model are the maximum daily temperature, the minimum daily temperature, and the time of occurrence of the maximum temperature. For the generation of a sinusoidal temperature profile the amplitude, period, offset, and phase of the sinusoid must be specified. For temperature variation the period is assumed to be 24 hours. The amplitude is assumed to be half of the difference between the maximum and minimum daily temperatures and the offset is assumed to be the average of maximum and minimum temperatures. The phase can be determined from the time of occurrence of the maximum temperature.

Rather than a mathematical sine function, the emulator model uses a table of 24 discrete points on a curve. The values in the table are normalized to lie between -1 and +1 with a zero offset. Values between the specified points are obtained by interpolation. The interpolated value is then scaled with the amplitude and offset to obtain the temperature.

3.3 Solar Energy

The effects of solar energy on the building are not included in the abridged model that was produced at NBS. The prediction of solar irriadiation on the building could be added to the model using the equations in the following sections.

3.3.1 Solar Radiation Intensity on the Building Wall

The total solar irriadiation on the outer surface of the building wall (including windows) is given by:

$$I = I_b \cos\theta + I_s + I_g \tag{3-1}$$

where I : total solar radiation
I_b : direct radiation
I_s : diffusive radiation
I_g : ground reflective radiation (assume $I_g = 0$)
θ : angle of incidence of the sun's rays

Direct and diffusive radiation are evaluated from the equations given by ASHRAE (1981), p. 27.2, and Walton (1983):

$$I_b = C_L A \exp[-B/\sin(\beta)] \tag{3-2}$$

$$I_s = C\, I_b \tag{3-3}$$

where C_L is a clearness number between 0.75 and 1.0, β is the solar altitude angle, and A, B, and C are constants. Monthly values are shown in Table 1 of ASHRAE (1981), p. 27.2. Walton (1983) provides a detailed procedure for obtaining A, B, and C for a given Julian date if desired.

3.3.2 Sine of Solar Altitude

$$\sin\beta = \sin(d)\sin(L) + \cos(d)\cos(L)\cos(h) \tag{3-4}$$

where d : declination angle (degrees) [ASHRAE (1981), p. 27.2]
L : latitude (degrees)
h : hour angle (degrees)
β : solar altitude angle (degrees)

The hour angle is given by:

$$h = 15\,(t_s - 12 + E/60 + TZN) - \lambda \tag{3-5}$$

where t_s : standard time of day (h)
E : equation of time (min) [ASHRAE (1981), p. 27.2]
λ : longitude (degree)
TZN : time zone number [Kusuda (1976), p. 9a]
= 4 for Atlantic, 5 for Eastern, 6 for Central, 7 for Mountain, 8 for Pacific

3.3.3 Cosine of Incident Angle

[ASHRAE (1981), p. 27.3]

$$\text{Cos } \theta = \cos(\beta) \cos(\phi + Y) \sin(v) + \sin(\beta) \cos(v) \tag{3-6}$$

where θ : angle of incidence of the sun's rays
ϕ : solar azimuth angle from south
Y : angle between the normal to the vertical surface and the south (facing angle)
v : tilt angle (the angle between the normal to the surface and the normal to the ground surface)
$\cos(\beta)$: cosine of solar altitude angle

$$= 1 - \sin^2 \beta \tag{3-7}$$

3.3.4 Sol-air Temperature

The abridged version of the emulator model program uses dry bulb outside air temperature for heat transfer between the wall and the outdoors. A more complete version would use sol-air temperature as given by the following equation for an opaque wall [Threlkeld (1970), p.342]:

$$T_{sol} = T_o + \frac{a\, I}{U_{ow}} \tag{3-8}$$

where a : solar absorptance of the exposed wall
I : solar radiation intensity (direct + diffuse)
T_o : outside air dry-bulb temperature
U_{ow} : thermal conductance of the wall outside of the mass center

4. AIR HANDLING UNIT MODEL

4.1 Assumptions

The air handling unit model uses steady state air enthalpy calculations to determine the amount of energy required for cooling and heating by the unit. Steady state equations are used since the time step used for the air handling unit model is similar in size to the time constants of HVAC systems. Possible EMCS control actions that provide input to the air handling unit model are starting and stopping the air handling unit, closing or allowing modulation of the outside air dampers, opening or closing minimum outside air dampers, and supply air temperature setpoint. The general model is designed to be configurable to represent either a constant or variable air volume (VAV) unit with either single or dual decks and heat exchangers for heating or cooling. The abridged model (section 2.0) that was implemented in the building emulator model computer program does not allow simulation of dual deck systems or systems with humidity control. See subroutine AHU in APPENDIX.

4.2 Equations

A sample of the model parameters for the air handling unit simulation subroutine was listed in table 2-9. The parameters in the table fall into two categories, which are configuration parameters and analog parameters. The configuration parameters are logical constants which are used to select a particular configuration for the air handling unit. There are four of these parameters, which are DUal-DECK (DUDECK), HUMidity-CONtrol (HUMCON), Return-Air-FAN (RAFAN), and Variable-Air-Volume (VAV), and which may be either true or false.

Figure 2-1 is a diagram of a generalized air handling unit showing the major components of the model. The two basic configurations for the air handling unit are dual-deck or single deck. To select dual deck, the logical parameter DUDECK is set true. If dual deck, there will be a set of heat exchangers for two paths and the supply fan will be located before the split. (Dual deck is not implemented in the abridged model) If the system is single deck, DUDECK will be false, the second set of heat exchangers will not be used, and the supply fan will be located after the last exchanger.

The decks in the model may contain a humidifier, a cooling coil, a heating coil or any combination of the three. The return air fan is optional and the logical parameter RAFAN determines whether it will be included in the model. If the parameter VAV is true, the air handling unit will be modeled as being Variable Air Volume (VAV), and the mass flow rate of air through the unit will be allowed to change to match the zone loading. If humidity control is to be used, parameter HUMCON must be set to true, and the supply air will be maintained at a setpoint for humidity as well as a dry bulb temperature setpoint (humidity control is not implemented in the abridged model).

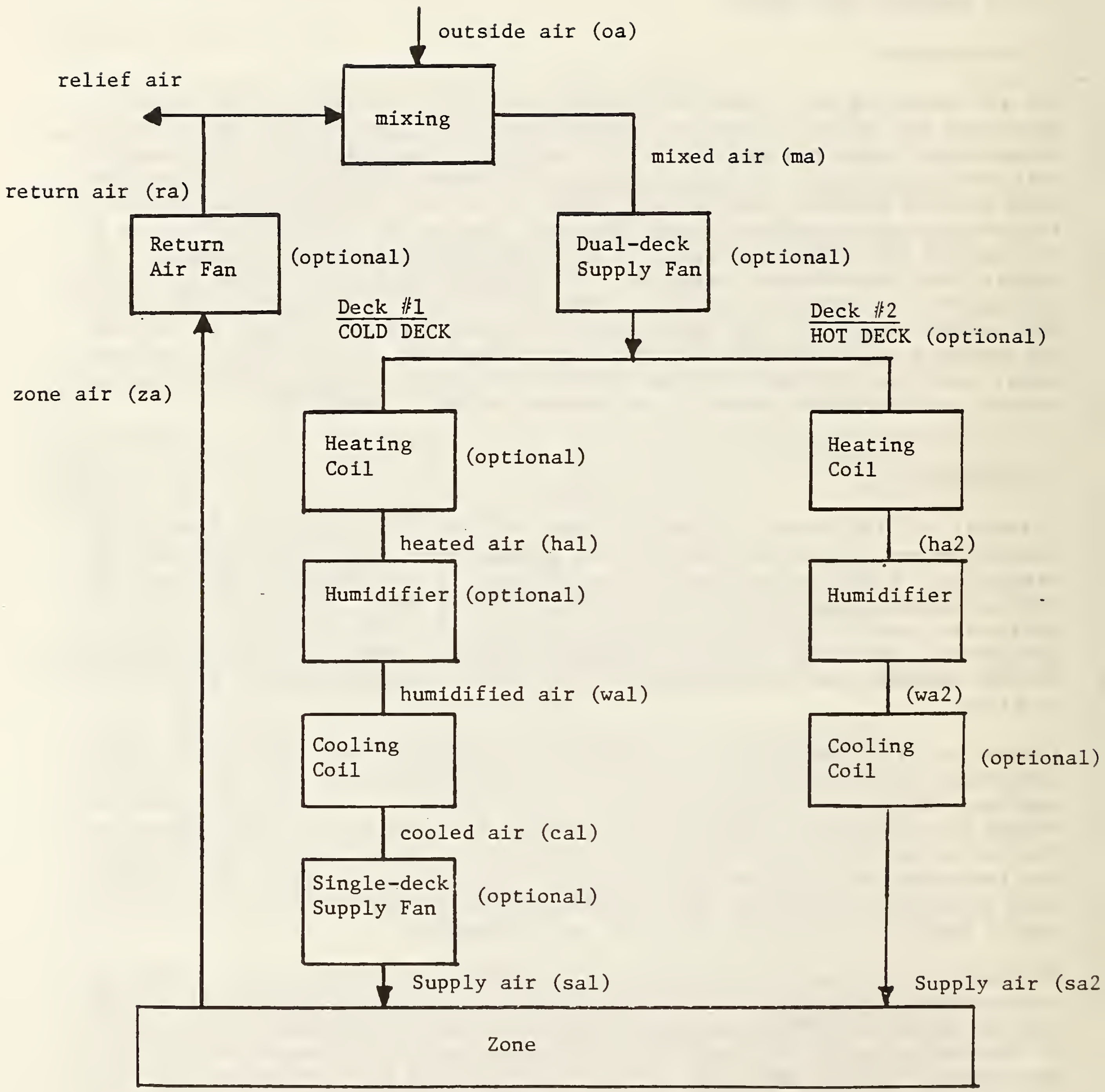

Figure 4-1. Block diagram of components of generalized HVAC system model

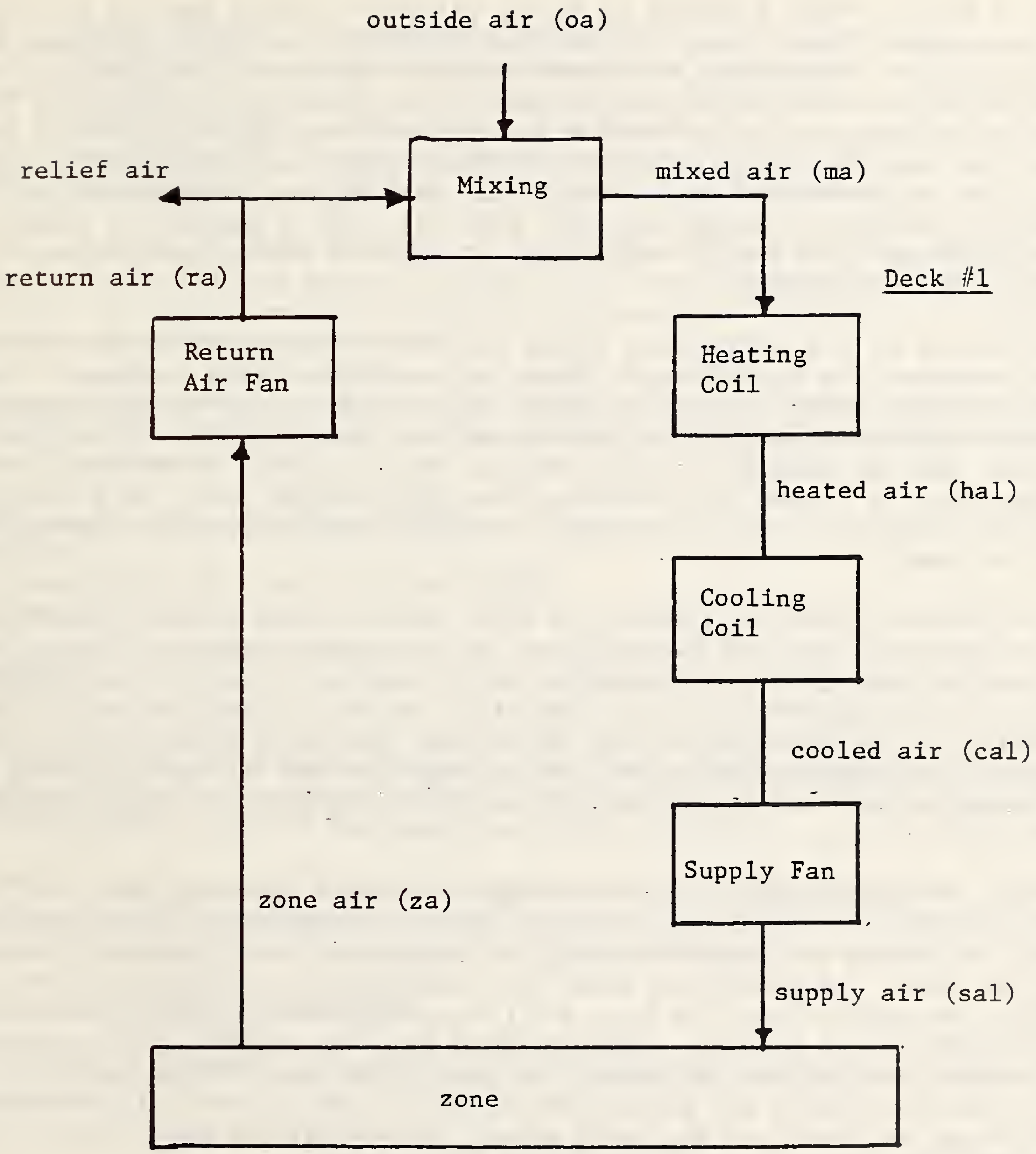

Figure 4-2. Block diagram of components of the abridged, single deck air handling unit model

Table 2-9 also shows a number of analog parameters which have to be specified for the model. These describe the capacities of the coils and fans in the unit. The fans in the unit will have a power requirement, a rated air volume flow rate, and a parameter to specify how much energy was added to the air passing through the fan by the fan waste heat. The fans for VAV will use the rated flow rate as the maximum but will have a flow rate and power consumption which can be lowered to match zone loads. Cooling and heating coils will have a certain capacity in energy per unit time that can be removed or added to the air in the unit. If a coil is given a capacity of zero, this indicates that the coil is not present.

In addition to the parameters there are inputs in the FORTRAN common area which contains the state vector. These are variables which represent commands from the EMCS which is being tested by the emulator. The variable ECONOmizer (ECONO) will be a signal to use or not use the outside air dampers for cooling. UNIT-ON (UNITON) will be true if the unit is to be running and false if the unit is turned off. VENtilation-OFF (VENOFF) will be false during occupied operation, but if the EMCS closes minimum outside air dampers VENOFF will be true.

The locations shown in figure 4-1 with two or three letter abbreviations beside them and location descriptions in parenthesis represent states of the air passing through the air handling unit. Each state can be characterized by the dry bulb temperature, T, the humidity ratio, W, the flow rate m and the enthalpy, h. The abbreviations on the diagram represent subscripts to variable names in the equations which are used to describe the air handling unit model. For example, the temperature of the air returning from the building zones is symbolized by T_{za}.

It will be assumed that if the capacity of the air handling unit will allow, the condition of supply air from the the air handler will be the same as specified by the air handler supply air setpoints. For a constant volume air handling unit, the supply air mass flow rate will be constant. If the system is VAV, the supply air flow rate will be determined by the local equipment model contained in the zone model (section 5.6). The return air flow rate will be assumed to equal the VAV supply air mass flow rate. If the system is dual-deck, the individual air mass flow rates for the hot and cold deck will be determined by the local equiment model. The return air mass flow rate will then equal the sum of the two deck mass flow rates.

The following FORTRAN functions are assumed for psychrometric calculations (the implementations of these functions are described in section 5.5). For the arguments listed, T is dry bulb temperature, W is humidity ratio, P is atmospheric pressure, RH is relative humidity, h is enthalpy, and DP is dewpoint temperature (also see Table 5-3).

1. h = ENTHAL(T,W,P) - enthalpy of moist air
2. T = TEMP(h,W,P)   - dry bulb temperature of moist air
3. DP = DEWPT(W,P)   - dew point temperature of moist air
4. W = HUMRAT(T,RH,P) - humidity ratio of moist air
5. RH = RELHUM(T,W,P) - relative humidity of moist air
6. T = TDBSAT(h,P)   - dry bulb temperature of saturated air

Figure 4-2 is a flow diagram showing the logic which determines the equations to use for calculating the states of the air in the air handling unit. The calculations for each time step are performed based on the return air state which is known from the zone air model.

Since the equations used for the air handling unit model depend on the state of several logical variables, a special format will be used to present the equations. The construction:

```
IF (condition1) is true (or false) THEN
    equation 1
ELSE IF (condition2) is true (or false) THEN
    equation 2
ELSE
    equation 3
ENDIF
```

will be used to denote logical decisions about which equation to use. Indentations will be used for clarity. See Figure 4-2 for logic flow and Table 4-1 for nomenclature of the equations which follow.

The following equations will apply if logical variable UNIT-ON is true. If it is not, then the air handling unit is off, no air is flowing, all mass flow rates will be zero, and all energy use by the air handler will be zero.

IF Variable-Air-Volume is true THEN

The supply air volume is assumed to be determined by the local zone VAV equipment.

ELSE

Constant volume is assumed. The mass flow rate m_{sa} equals the rated air flow rate.

ENDIF (if Variable-Air-Volume is true).

IF Return-Air-FAN is true THEN

Assume that the return air fan causes a constant amount of thermal energy to be added to the return air per second. The return air fan heating rate must be provided as a parameter, q_{rf}. Then the temperature rise due to the return air fan is:

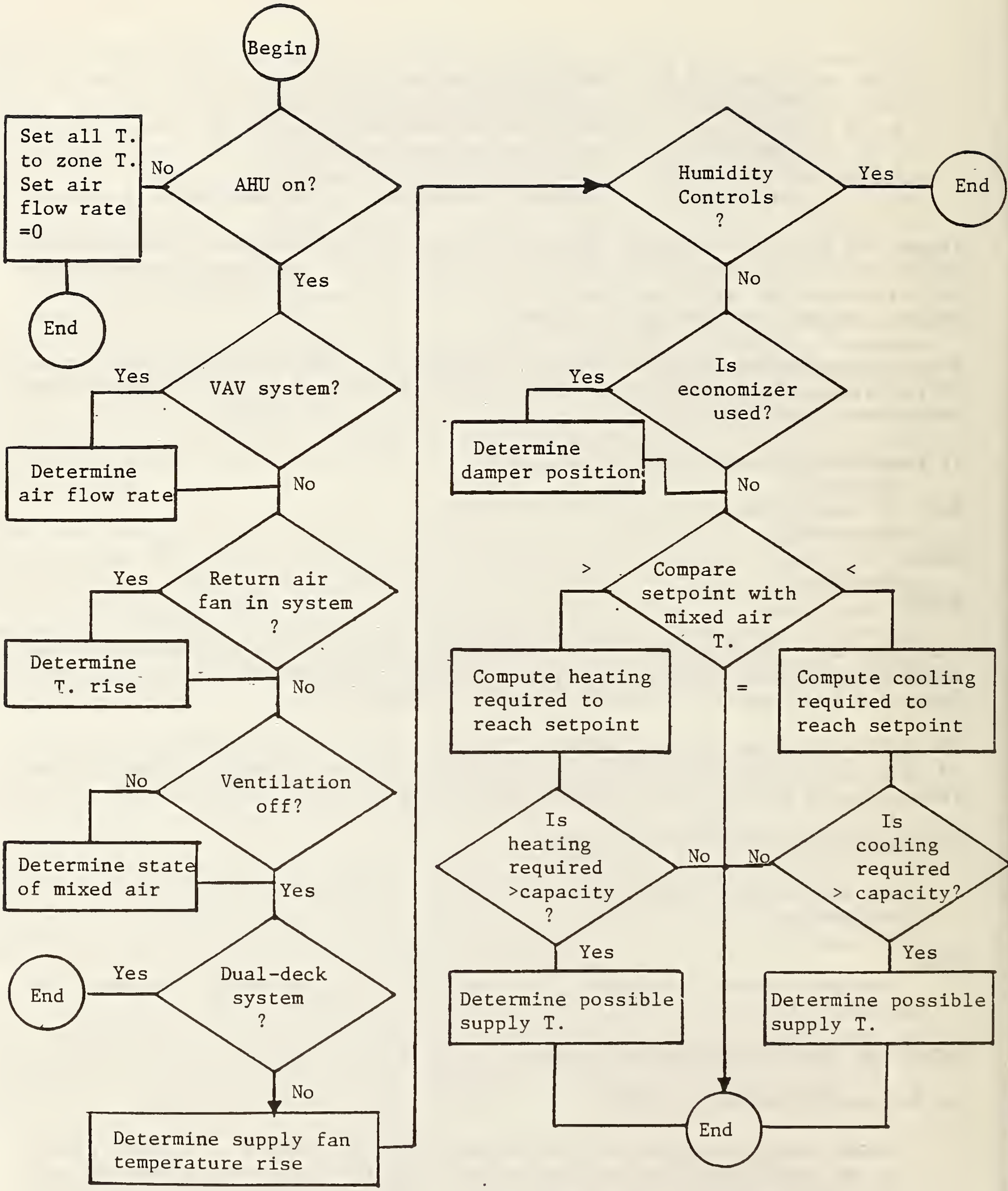

Figure 4-3. Air handling unit model calculation sequence

$$dT_{rf} = \frac{q_{rf}}{m_{ra}\ c_{pa}} \tag{4-1}$$

$$\text{and } T_{ra} = T_{za} + dT_{rf} \tag{4-2}$$

ELSE

$$T_{ra} = T_{za} \tag{4-3}$$

ENDIF (if Return-Air-FAN is true)

$$W_{ra} = W_{za} \tag{4-4}$$

$$h_{ra} = \mathrm{ENTHAL}(T_{ra}, W_{ra}, P) \tag{4-5}$$

The next calculations determine the state of the mixed air (subscript mc) if the outside air dampers are closed. It will be assumed that f_{va} is a model parameter which the fraction of air mass flow through the air handling unit which is the minimum outside air flow rate for ventilation purposes. If the EMCS cuts off ventilation air as an algorithm action, then this value will be zero:

IF VENtilation-OFF is true THEN

$$h_{mc} = h_{ra} \tag{4-6}$$

$$W_{mc} = W_{ra} \tag{4-7}$$

$$T_{mc} = T_{ra} \tag{4-8}$$

ELSE

$$h_{mc} = f_{va}\ h_{oa} + (1 - f_{va})\ h_{ra} \tag{4-9}$$

$$W_{mc} = f_{va}\ W_{oa} + (1 - f_{va})\ W_{ra} \tag{4-10}$$

$$T_{mc} = \mathrm{TEMP}(h_{mc}, W_{mc}, P) \tag{4-11}$$

ENDIF (if VENtilation-OFF is true)

IF DUal-DECK is false THEN

For a single deck air handling unit, the supply fan temperature rise is calculated as the return fan rise was calculated:

Table 4-1. Nomenclature for the air handling unit model

c_{pa}	specific heat of air
DP	dew point
dT	temperature rise
f_{va}	fraction of air mass flow for the minimum outside air for ventilation
h	enthalpy
m	air mass flow rate
P	barometric pressure
q	heating/cooling rate
$q_{c,max}$	cooling capacity
$q_{h,max}$	heating capacity
RH	relative humidity
T	air temperature
T_{set}	setpoint temperature
W	humidity ratio

Subscripts:

c	cooling
ca	cooled air
h	heating
ha	heated air
ma	mixed air
mc	state of mixed air if the outside air dampers are closed
mo	state of mixed air if the outside air dampers are completely open
oa	outside air
ra	return air
rf	return air fan
sa	supply air
sf	supply air fan
wa	humidified air
za	zone air

$$dT_{sf} = \frac{q_{sf}}{m_{sal}\ c_{pa}} \tag{4-12}$$

IF HUMidity-CONtrol is false THEN

If there is no humidity control, there will only be a dry bulb temperature setpoint. The humidity will be allowed to float to any level.

IF ECONOmizer is true THEN

The EMCS allows the air handling unit to use the outside air for conditioning supply air. No provision is made in this version of the model for direct EMCS control of damper position. It is assumed that local controls determine proper outside air damper position. However, once the outside air damper is fully open when cooling is required, it remains open even if outside air enthalpy is high. The EMCS must take responsibility for closing the damper in this case. To determine the outside air damper position:

$$h_{mo} = h_{oa} \, , \tag{4-13}$$

$$T_{mo} = T_{oa}, \text{ and} \tag{4-14}$$

$$W_{mo} = W_{oa} . \tag{4-15}$$

where the mo subscript indicates the state of the mixed air if the outside air dampers are completely open.

If $T_{mc} \leq T_{set} - dT_{sf}$, this indicates that the outside air damper should be closed and that heating of the supply air will be required. In this case:

$$T_{ma} = T_{mc} \quad , \tag{4-16}$$

$$W_{ma} = W_{mc} \quad \text{and} \tag{4-17}$$

$$h_{ma} = h_{mc} \, . \tag{4-18}$$

However, if $T_{mc} > T_{set} - dT_{sf}$ and $T_{mo} > T_{set} - dT_{sf}$, this indicates that that the outside air damper should be wide open, but additional cooling of supply air will be needed using a cooling coil. In this case:

$$T_{ma} = T_{mo} \quad , \tag{4-19}$$

$$W_{ma} = W_{mo} \quad \text{and} \tag{4-20}$$

$$h_{ma} = h_{mo} \, . \tag{4-21}$$

But, if $T_{mc} > T_{set} - dT_{sf}$ and $T_{mo} \leq T_{set} - dT_{sf}$, then the outside air dampers will be able to provide all conditioning of the air passing through the air handler. In this case, the outside air dampers will be at some intermediate position. The supply air temperature will equal the setpoint but since the fraction of outside air is not known in this case, the exact psychrometric state of the supply air (and thus the enthalpy) is not known. Theoretically, since the dry-bulb temperature lines on the psychrometric chart are not parallel, an iterative technique would be required to determine W_{ma} and h_{ma}. However, the lines are approximately parallel and an approximate relationship can be used since the state of the mixed air lies on a straight line between the state of the air with the dampers fully open and the state with the dampers closed. This is described by

$$\frac{T_{ma} - T_{mo}}{T_{mc} - T_{mo}} = \frac{W_{ma} - W_{mo}}{W_{mc} - W_{mo}} \quad \text{or} \tag{4-22}$$

Then, since: $T_{ma} = T_{set} - dT_{sf}$, (4-23)

$$W_{ma} = \frac{W_{mc} - W_{mo}}{T_{mc} - T_{mo}} (T_{set} - dT_{sf} - T_{mo}) + W_{mo} \tag{4-24}$$

$$h_{ma} = \mathrm{ENTHAL}(T_{ma}, W_{ma}, P) \tag{4-25}$$

ELSE

If the EMCS keeps the outside air dampers closed:

$$T_{ma} = T_{mc} \quad , \tag{4-26}$$

$$W_{ma} = W_{mc} \quad \text{and} \tag{4-27}$$

$$h_{ma} = h_{mc} \ . \tag{4-28}$$

ENDIF (ECONOmizer is true)

If outside air is providing all of the conditioning, as determined above ($T_{ma} = T_{set} - dT_{sf}$), then no heating or cooling of the mixed air is required, and:

$$T_{cal} = T_{ma} \ , \tag{4-29}$$

$$W_{cal} = W_{ma} \ , \tag{4-30}$$

$$h_{cal} = h_{ma} \ , \tag{4-31}$$

$$T_{hal} = T_{ma} \ , \tag{4-32}$$

$$W_{hal} = W_{ma} \ , \text{ and} \tag{4-33}$$

$$h_{hal} = h_{ma} \ . \tag{4-34}$$

However, if heating of the mixed air is required ($T_{ma} < T_{set} - dT_{sf}$) then the air must be heated from T_{ma} to T_{hal}, where:

$$T_{hal} = T_{set} - dT_{sf}. \tag{4-35}$$

The simplified psychrometric process for air heating is to increase the temperature of the air at constant humidity ratio (no moisture is added). Therefore:

$$W_{hal} = W_{ma} \quad \text{and} \tag{4-36}$$

$$h_{hal} = \mathrm{ENTHAL}(T_{hal}, W_{ma}, P). \tag{4-37}$$

Then the energy added to the air is:

$$q_h = m_{sal}(\, h_{hal} - h_{ma}\,). \tag{4-38}$$

If q_h is greater than the heating capacity of the air handling unit, then the supply air temperature will be below the supply air setpoint. The supply air temperature is calculated as follows:

$$h_{hal} = \frac{q_{h,max}}{m_{sal}} + h_{ma} \quad , \text{and} \tag{4-39}$$

$$T_{hal} = T(h_{hal}, W_{ma}, P) \quad . \tag{4-40}$$

Then the energy added to the air is:

$$q_h = q_{h,max} \tag{4-41}$$

Alternatively, cooling of the air may be required ($T_{ma} > T_{set} - dT_{sf}$). Then the air must be cooled from T_{ma} to T_{cal}, where:

$$T_{cal} = T_{set} - dT_{sf}. \tag{4-42}$$

The assumed psychrometric process for air cooling is to remove energy from the air at constant humidity ratio until the dew point temperature of the mixed air is reached, at which point energy and water are removed from the air, which remains saturated, until T_{cal} is reached.

The dewpoint temperature must be known for the calculations:

$$DP_{ma} = \mathrm{DEWPT}(W_{ma}, P) \tag{4-43}$$

If $T_{cal} > DP_{ma}$, then the supply air will not be at a saturated condition after it is cooled and:

$$h_{cal} = h(T_{cal}, W_{ma}, P) \text{ , and} \tag{4-44}$$

$$W_{cal} = W_{ma} \text{ .} \tag{4-45}$$

However, if $T_{cal} \leq DP_{ma}$ then condensation of moisture from the air will occur and the supply air will be assumed to be saturated:

$$RH_{cal} = 100 \text{ \% ,} \tag{4-46}$$

$$W_{cal} = HUMRAT(T_{cal}, RH_{cal}, P) \text{ , and} \tag{4-47}$$

$$h_{cal} = ENTHAL(T_{cal}, W_{cal}, P). \tag{4-48}$$

Then the cooling energy required to maintain the setpoint is:

$$q_c = m_{sal}(h_{ma} - h_{cal}) \tag{4-49}$$

If q_c is larger than the rated capacity of the cooling coil, then the supply air state will be above the setpoint of the air handling unit. The supply air state and cooling energy will then have to be calculated as follows:

$$h_{cal} = h_{ma} - \frac{q_{c,max}}{m_{sal}} \text{ , and} \tag{4-50}$$

$$q_c = q_{c,max} \tag{4-51}$$

If $h_{cal} > ENTHAL(DP_{ma},W_{ma},P)$, then the supply air at maximum cooling capacity is not saturated:

$$T_{cal} = TEMP(h_{cal}, W_{ma}, P)\text{, and} \tag{4-52}$$

$$W_{cal} = W_{ma}. \tag{4-53}$$

However, if $h_{cal} \leq h(DP_{ma},W_{ma},P)$, then the supply air at maximum cooling capacity is saturated:

$$T_{cal} = TDBSAT(h_{cal}, P)\text{, and} \tag{4-54}$$

$$W_{cal} = HUMRAT(T_{cal}, RH{=}100\%, P) \tag{4-55}$$

The final supply air state is then:

$$W_{sal} = W_{cal} \tag{4-56}$$

$$T_{sal} = T_{cal} + dT_{sf} \tag{4-57}$$

$h_{sa1} = \mathrm{ENTHAL}(T_{sa1}, W_{sa1}, P)$ (4-58)

ELSE

If it is in the heating season and the humidity is too low, a humidifier can be used to increase the humidity ratio of the supply air. If it is in the cooling season and humidity is excessive, the air can be dehumidified by overcooling and reheating of air. Humidity control was not implemented in the abridged version of the emulator model and equations will not be presented.

ENDIF (if HUMidity-CONtrol is false)

ELSE

Equations for dual deck were not implemented in the abridged version of the emulator model and will not be presented.

ENDIF (if DUal-DECK is false)

The states of the air passing through the air handling unit are stored in the state vector and are made available to other model subroutines in the building emulator model program.

5. BUILDING SPACE ZONE MODEL

The building zone model subroutine, unlike the other model equation subroutines, implements a model formulated in terms of differential equations. A dynamic rather than a steady-state model is necessary since the time step magnitude used in the model (smaller than one minute) is much smaller than the characteristic time constants of the zone.

5.1 Assumptions

A number of simplifying assumptions are made in order to allow a relatively simple set of model equations. The building zone to be modeled is assumed to be a typical perimeter zone of a large multi-story building as Kao (1983) used in his energy conservation study for a large office building. The zone is enclosed by an exposed wall with a glass window, three interior walls, ceiling, and floor. Since there is one exposed wall and one glass window the governing equations are simplified. The following assumptions are made for the building zone model:

(1) Uniform temperature distributions on surfaces.

(2) No stratification of zone air (fully mixed air).

(3) No heat or mass flow between zones.

(4) Simple, uniform optical and thermal properties for any glass windows.

(5) The term 'wall' includes walls, ceiling, and floor.

(6) All wall structures (including ceiling and floor) which are not exposed directly to the outside are treated as thermally adiabatic. (No conductive heat flow across the wall structure is assumed).

(7) Simple two-node representation of the exterior wall with one node at the interior wall surface and one node at the mass center point of the wall.

(8) Convective and radiative heat transfer coefficients for the inner surfaces of walls are considered constant.

(9) The surface heat transfer coefficient of outside wall surfaces is considered constant and is not a function of outdoor conditions (including wind).

(10) The wind speed is considered constant and has an influence only on infiltration.

(11) All static pressures in the HVAC system, zone and outdoor air are assumed to be constant with a value of 29.92 inches of mercury.

(12) The latent heat gain from furnishings in the zone is neglected.

(13) The heat capacity of zone furnishings is assumed to remain the same at all times.

(14) Radiant heat from lights (short wave radiation) is assumed to be uniformly distributed on all inner surfaces of the zone.

(15) For all glass windows, the same constant values of transmittance and shading coefficient are used.

(16) For all exposed opaque walls, solar absorptance is a constant.

(17) Solar radiation intensity consists of direct and diffuse radiation. Ground reflective radiation is ignored. The solar constants A, B, and C appearing in equations (3-2) and (3-3) are assumed constant for a given specific month.

(18) It is assumed that solar radiation entering the zone through windows is not absorbed by glass but is distributed uniformly on the inner surfaces of the zone.

Based on these assumptions, it is assumed that five state variables are sufficient to model the zone. The state variables are room air dry bulb temperature, room humidity ratio, exterior wall interior surface temperature, glass interior surface temperature, and exterior wall internal temperature.

The abriged model (see section 2) implemented in the building emulation simulation program included some additional assumptions. The equations presented in the following sections will usually be for a complete model. The changes to equations for the abridged model will be described whenever necessary in the following sections. The assumptions for the abridged model are:

1) No effects of solar energy.

2) No radiative heat transfer between zone surfaces.

5.2 Fundamental Equations

Since there are five state variables in the building emulator, five equations relating to these variables are required to solve for the values of the variables. Table 5-1 contains nomenclature for the fundamental equations of the zone model.

5.2.1 Zone Air Heat Balance

The first fundamental equation for the emulator zone model expresses the balance in heat transfer between the air and furnishings in the zone and the wall and window surfaces, entering supply air from the air handling unit, infiltration heat transfer, and internal sources of heat [Borresen (1981), Modest and Macken (1977), Bullock (1984), Park (1981)]. The equation is:

$$C_{zone} \frac{dT_i}{dt} = Q_{supply} + Q_{infilt} + Q_{convec} + Q_{internal} \tag{5-1}$$

The left hand side of the equation represents the storage of energy in the zone air and furnishings. The state variable is the interior air temperature. Of the four terms on the right side of the equation, the energy from the supply air and the internal gains may be considered boundary conditions. The supply air condition is determined by the air handling unit model and the internal gain, consisting of heat gains from occupants, lights, and equipment, is determined by the building schedule. The other two terms for convection and infiltration heat transfer are functions of the other state variables.

Table 5-1. Nomenclature for equations in the building space zone model

State Variables:

T_i	- space dry bulb temperature	W_i	- space humidity ratio
T_w	- wall mass center temperature	T_{ig}	- window glass temperature
T_{iw}	- wall interior surface temperature		

Other Parameters:

A	- conduction coefficient	B	- convection coefficient
C	- thermal mass	T	- temperature
M_{air}	- air mass for water absorption	Q	- heat transfer rate
T_{mr}	- space mean radiant temperature	R	- radiative coefficient
G	- zone moisture increase rate		
S_{gw}	- wall heat gain by solar radiation through windows		

Subscripts:

convec - convection	o - outside air	g - glass
infilt - infiltration	w - at the wall mass center	

5.2.2 Exterior Wall Mass Heat Balance

The exterior wall is assumed to be the only surface enclosing the zone which exchanges energy with the outdoor environment and the indoor environment. The state variable for wall temperature is assumed to be the temperature at the mass center of the wall. The mass center is defined as that point in the wall where for a unit area of wall the total mass of the wall outside of the mass center is equal to the total mass of the wall inside the mass center. For a masonry construction, this may result in the mass center being different from the geometric center of the wall. The equation for the exterior wall is:

$$C_{wall} \frac{dT_w}{dt} = Q_{inside} + Q_{outside} + Q_{solar} \tag{5-2}$$

The left hand side of the equation represents the storage of energy in the building wall. The first term on the right is the heat transfer between the wall interior and the inner wall surface. This term is dependent on state variables. The second two terms on the right represent the heat transfer between the wall interior and the outdoor environment, including the impingement of solar irradiation on the outside of the wall. The right two terms are time-dependent boundary conditions.

5.2.3 Zone Air Moisture Balance

The air in the zone is assumed to contain moisture. The state variable used to describe the water content of the air is the humidity ratio. The equation used to describe the moisture balance is [Chi (1982), Bullock (1984)]:

$$M_{air} \frac{dW_i}{dt} = G_{infilt} + G_{supply} + G_{internal} \tag{5-3}$$

The left hand side of this equation represents the storage of water vapor in the air and furnishings of the zone. The right hand side of the equation has three terms. The first term is the moisture transfer with the outdoor environment due to infiltration, which contains a state variable. The second term is the net moisture added to or removed from the room by the air handling unit. This is the difference in absolute moisture between the supply and return air from the air handling unit, and is a boundary condition. The third term is the moisture generated internally by occupants and equipment. The deficiency in this equation is that it assumes the air and furnishings can absorb an infinite amount of moisture when in reality, the air can become saturated. The emulator model zone subroutine performs a check of the relative humidity of the air to prevent oversaturation.

5.2.4 Inner Wall Surface Heat Balance

For the inner wall surface of the exterior wall, the heat balance is expressed by an algebraic equation rather than a differential equation since there is no storage of energy at the wall surface. The equation is:

$$A_w(T_w - T_{iw}) + B_w(T_i - T_{iw}) + R_w(T_{mr} - T_{iw}) + S_{gw} = 0 \qquad (5\text{-}4)$$

This equation is based on the assumption that all heat transfer in and out of the node at the wall surface sums to zero. The first term is the conduction heat transfer between the wall interior and the inner wall surface. A_w is a conduction coefficient. The second term is the convection heat transfer between the wall surface and the zone mean radiant temperature. The final term is due to any solar irradiated energy which passes through the zone window and is absorbed by the zone inside surfaces.

5.2.5 Inner Glass Surface Heat Balance

For the inner glass surface of the zone window, if present, an algebraic equation describes the heat transfer balance since there is no storage of energy:

$$A_g(T_o - T_{ig}) + B_g(T_i - T_{ig}) + R_g(T_{mr} - T_{ig}) = 0 \qquad (5\text{-}5)$$

This equation expresses that the sum of all heat transfer in and out of the node at the glass surface is zero. The first term is the conduction/convection heat transfer between the inner glass surface and the outdoor air. The coefficient A_g is a combination convection and conduction coefficient. The second term in the equation is the convection heat transfer between the glass surface and the interior air. The coefficent B_g is for convection. Radiant heat transfer between the glass surface and the mean radiant temperature of the room is represented by the third term of the equation.

5.3 Solution of Equations

The equations described in section 5.2 are solved by the zone model subroutine using a simple explicit forward difference numerical technique (Euler method) to predict the values of the state variables one step forward in time. The time step used within the zone model is different from the emulator time step used for the other models in the emulator simulation program. Since simple differential equation solution technique is used within the zone model, the time steps must be small or the solution will not converge. The emulator time step is divided into a number of smaller zone model time steps and the differential equations are solved for each zone model time step until the emulator time step has passed. However, the zone model is not constrained to execute zone model time steps in real time.

The subroutine ZONE performs the steps required to determine the values of the state variables at the next emulator time step. This routine first determines the best value for the zone model time step. This process is described in section 5.3.2. Then ZONE calls the two subroutines EULER and SURFAC in sequence, and advances the time by one zone model time step until the emulator time step has passed. The subroutine EULER is used to solve the three differential equations listed above (5-1 through 5-3) for one zone model time step. SURFAC determines the wall and glass surface temperatures for the current time step using the algebraic equations described above (5-4 through 5-5).

5.3.1 Euler Forward Difference Technique

The subroutine EULER is a generalized routine to advance a system of ordinary first order differential equations by one time step using the explicit difference method. The arguments to the routine include the number of differential equations, an array containing the dependent (state) variables, the time step, and the name of a FORTRAN function which will evaluate the right hand sides of the differential equations when they are expressed in the form:

$$\frac{dY_i}{dt} = f\ (t,\ Y_1,\ Y_2,\ \ldots Y_n),\ i=1,2,\ldots,n \tag{5-6}$$

where Y_i is the ith dependent variable and n is the number of equations. The value of Y at time t+ t can approximately be predicted by:

$$Y_i(t+\Delta t) = Y_i(t) + \Delta t * \text{FUNCTION}\ (i,t,n,Y_1,Y_2,\ldots Y_n) \tag{5-7}$$

As long as the product of the time step Δt and the function value is small, a solution to the differential equations can be obtained.

5.3.2 Adaptive Time Step

The time step used by the EULER subroutine is determined adaptively to become smaller when the zone is undergoing rapid changes in the state variables and larger as the zone approaches steady state. Subroutine ZONE estimates a new time step each time it is called. The estimation is based on the requirement that the change in any state variable for one Euler time step must not exceed a selected maximum value. For example, in the building emulator model program, this maximum has been chosen to be 0.1 degrees F for temperature state variables. Since the value of the function which describes the right hand side of the differential equation (described in section 5.3.1) is the rate of change of the state variable, the smallest desirable zone model time step may be determined by dividing the maximum allowable change in the state variable per time step by the current value of the right hand side function. Of course, calculations must be performed to ensure that the zone model time step is not larger than the emulator time step or smaller than a certain minimum. In the building emulator program zone model subroutine the time step can be no smaller than 0.001 seconds.

After the zone model time step is determined, it is adjusted so that it evenly divides the emulator time step. The quotient of this division is the number of times that the ZONE subroutine must call the EULER and SURFAC subroutines for each emulator model time step.

5.3.3 Differential Equation Function

The FORTRAN function ZONEFN is used by the zone model subroutine to calculate the current value of the right hand sides of the differential equations describing the state variables. Since there are three differential equations for the zone model, there are three possible right hand sides of the differential equations. The right hand side to be calculated is determined by an index which is 1 for the zone air energy balance, 2 for the zone moisture balance, and 3 for the wall energy balance. The expanded right hand sides of the equations corresponding to equations (5-1 through 5-3) are presented below. The variables which are designated by an upper case F with a subscript are implemented in the ZONEFN subroutine in a special way. Rather than being variables or constants, they are FORTRAN functions which return the value of a coefficient or energy term. In the definition of terms following each equation below, the FORTRAN function name used in the emulator model program will be listed. For the zone air energy balance:

$$\mathrm{ZONEFN} = \frac{F_{sa} + F_I(T_o - T_i) + F_{cw}(T_{iw} - T_i) + F_{cg}(T_{ig} - T_i) + F_{ig}}{F_{cz}} \qquad (5\text{-}8)$$

where: F_{sa} : energy entering or removed from the zone by the air handling unit (function SUPPLY).

F_I : coefficient for infiltration of outdoor air into zone (function INFILT).

F_{cw} : coefficient for convection between the exterior wall surface and the zone air (function CONVWA).

F_{cg} : coefficient for convection between the surface of a window in the exterior wall and the zone air (function CONVGL).

F_{ig} : energy entering the zone due to internal gains from lights, equipment, and people (function INTERN).

F_{cz} : thermal mass of the zone air and furnishings (function MASSTH).

For the zone moisture balance:

$$\text{ZONEFN} = \frac{F_{sw} + F_{Im}(W_o - W_i) + F_{mg}}{F_{mz}} \tag{5-9}$$

where: F_{sw} : moisture added to or removed from the zone by the air handling unit (function SUPPLW).

F_{Im} : coefficient for infiltration of moisture (or exfiltration) from outdoor air into zone (function INFILW).

F_{mg} : moisture entering the zone due to internal moisture gain from equipment and people (function MOIST).

F_{mz} : effective mass of zone air and furnishings which can absorb moisture (function MASSA).

for the prediction of the rate of change of moisture in the zone, an additional check is made to insure that the zone air does not become oversaturated. This is accomplished by first using the function HUMRAT to determine the humidity ratio at the zone air temperature, T_i, and 100 percent relative humidity. If the current indoor humidity ratio, W_i, exceeds this value and ZONEFN is a positive value, then ZONEFN is set to zero, indicating that no further increase in zone moisture is possible. If W_i is zero and ZONEFN is negative, ZONEFN is set to zero indicating that zone moisture cannot be decreased since the air in the zone is completely dry.

The right hand side function for the exterior wall mass heat balance is:

$$\text{ZONEFN} = \frac{F_{co}(T_{sol} - T_w) + F_{ci}(T_{iw} - T_w)}{F_{cw}} \tag{5-10}$$

Where: F_{co} : coefficient for conduction and convection heat transfer between the mass center of the wall and the outdoor air (function CNDCTO).

F_{ci} : coefficient for conduction heat transfer between the mass center of the wall and the inner surface of the wall (function CNDCTI).

T_{sol} : the sol-air temperature (defined in section 3.3.4)

F_{cw} : thermal mass of the wall structure (function MASSW).

5.3.4 Surface Heat Balance Subroutine

The subroutine SURFAC is called by ZONE to solve the algebraic equations (5-4) and (5-5) for the inner surface temperatures of the exterior wall and glass window. The two equations after rearrangement to solve for the surface temperatures are shown below. The variables which are designated by an upper case F with a subscript are implemented in the SURFAC subroutine in a special way. Rather than being variables or constants, they are FORTRAN functions which return the value of a coefficient or energy term. In the definition of terms following each equation below, the FORTRAN function name used in the emulator model program will be listed. For the exterior wall inner surface balance:

$$T_{iw} = \frac{F_{Aw}\,T_w + F_{Bw}\,T_i + F_{Rw}\,F_{Tr} + F_{ts}}{F_{Aw} + F_{Bw} + F_{Rw}} \qquad (5\text{-}11)$$

Where: F_{Aw} : coefficient for conduction heat transfer between inner surface of wall and mass center of wall (function CNDCTI).

F_{Bw} : coefficient for convection heat transfer between inner surface of wall and zone air (function CONVWA).

F_{Rw} : coefficient for radiation heat transfer between inner surface of wall and zone radiant temperature (function WRADIA).

F_{Tr} : mean radiant temperature of zone (function TRN).

F_{ts} : energy which passes through windows in the zone and is absorbed at the wall surface (function WSOLAR).

For the window glass inner surface heat balance:

$$T_{ig} = \frac{F_{Ag}\,T_o + F_{Bg}\,T_i + F_{Rg}\,F_{Tr}}{F_{Ag} + F_{Bg} + F_{Rg}} \qquad (5\text{-}12)$$

Where: F_{Ag} : coefficient for conduction and convection heat transfer between the inner surface of the glass and the outdoor air (function GCNDCT).

F_{Bg} : coefficient for convection heat transfer between the inner surface of the glass and the zone air (function GCNVCT).

F_{Rg} : coefficient for radiation heat transfer between the inner surface of the glass and the zone radiant temperature (function GRADIA).

F_{Tr} : mean radiant temperature of zone (function TRN).

5.3.5 Fortran Coefficient Functions

The names of the special FORTRAN functions used to give coefficients or terms in the equations (5-8) through (5-12) were listed in the definitions following the equations. All of these functions have the same general form. The functions can be designated as either time-dependent or time-independent. Values of coefficients such as conduction coefficients may be considered time-independent and will need to be calculated only once. In this case, the function automatically determines if the function has already been evaluated and if so, returns the previously determined value.

Other coefficients such as a coefficient for infiltration heat transfer must usually be considered time-dependent. In this case the coefficient will be calculated each time the function is called unless the function is called twice for the same time step. In this case the function returns the previously determined value.

The general form of the FORTRAN function for a coefficient function is shown in table 5-1. The parameter TIMEDP is set to true for a time-dependent function and false for a time-independent function. The specific equations to evaluate the functions used in the emulator model program are listed in section 5.4.

5.4 Equations for Heat and Mass Transfer Coefficients and Terms

The following sections contain the equations used to evaluate the terms and coefficients to expand equations (5-8) through (5-12). The name of the FORTRAN coefficient function (see section 5.3.5) will be given in parenthesis following the title of the section. The equations listed would be placed in the area marked 'equations to evaluate function' in the generalized coefficient function listing in table 5-2.

Table 5-2. General form of the FORTRAN function for a coefficient function

```
      FUNCTION [name](TIME)

      LOGICAL TIMEDP, NOINIT
      PARAMETER(TIMEDP = .TRUE.)
      INTEGER OLD,TIME
      SAVE NOINIT,[name],FUNC
      DATA NOINIT/.TRUE./,OLD/-1/

      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
        FUNC = [ equations to evaluate function ]
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      [name] = FUNC
      RETURN
      END
```

5.4.1 Sensible Heat Gain/Loss from Supply Air (SUPPLY)

The energy added to or removed from the zone by the supply air is determined from the difference in enthalpies of return and supply air. In addition, the energy which is added to or removed from the room by the local zone heating or cooling equipment is included. The equation used is:

$$F_{sa} = h_{sa1}\, m_{sa1} + h_{sa2}\, m_{sa2} - h_{za}\, m_{ra} + Q_{\ell h} - Q_{\ell c} \quad (5\text{-}13)$$

Where: h_{sa1} : enthalpy of supply air
m_{sa1} : mass flow rate of supply air
h_{sa2} : enthalpy of hot deck supply air (optional)
m_{sa2} : mass flow rate of hot deck supply air (optional)
h_{za} : enthalpy of zone air
m_{ra} : mass flow rate of return air
$Q_{\ell h}$: energy added to zone by local zone heating equipment
$Q_{\ell c}$: energy removed from zone by local zone cooling equipment

This function is considered to be time-dependent.

5.4.2 Coefficient for Sensible Heat Gain/Loss due to Infiltration (INFILT)

Sensible gain/loss due to infiltration is assumed to equal a coefficient multiplied by the difference in temperature between zone and outdoor air. The coefficient is determined from:

$$F_I = C_{p,infl}\, \rho_{infl}\, V_i\, I_{air} \quad (5\text{-}14)$$

where $C_{p,infl}$: specific heat of infiltrated air
ρ_{infl} : density of infiltrated air
I_{air} : air exchange rate (air changes/second)
V_i : volume of zone space

The air exchange rate is considered to be a function of windspeed and indoor-outdoor temperature difference [Kusuda and Saitoh (1980), Achenbach and Coblenz (1963)]. The equation for air exchange rate per hour is:

$$I_{air} = \frac{I_{s,air}}{0.695} \left[0.15 + 0.013\, V_w + 0.005\, |T_o - T_i|\right] \quad (5\text{-}15)$$

where V_w : wind speed (mph)
$I_{s,air}$: standard air exchange rate (model parameter, see table 2-8)
= 1.5 for leaky building
1.0 for standard building
0.5 for modern type building

This function is considered to be time dependent.

5.4.3 Coefficient for Convective Heat Flow Rate from Walls (CONVWA)

The convective heat flow rate from the inner surface of the zone walls, including floor and ceiling, is assumed to be equal to a coefficient multiplied by the difference between the wall surface temperature and the zone air temperature. The coefficient is given by:

$$F_{cw} = h_{iw,c} A_w \tag{5-16}$$

where $h_{iw,c}$: convective heat transfer coefficient at the interior surface (model parameter, see table 2-8)

A_w : surface area of the wall (model parameter, see table 2-8).

This coefficient is assumed to be time-independent.

5.4.4 Convective Heat Flow Rate from Glass Windows (CONVGL)

The convective heat flow rate from the inner glass surface of zone windows is assumed to be equal to a coefficient multiplied by the difference between the glass surface temperature and the zone air temperature. The coefficient is given by:

$$F_{cg} = h_{ig,c} A_g \tag{5-17}$$

where $h_{ig,c}$: convective heat transfer coefficient of inner glass surface to zone air

A_g : the glass window area

This coefficient is assumed to be time-independent. The abridged zone model effectively assumes that there are no windows in the zone, and therefore this coefficient is not used.

5.4.5 Sensible Internal Gains (INTERN) and Latent Internal Gains (MOIST)

5.4.5.1 Internal Gain Schedule Table

The FORTRAN functions used to determine internal gains, INTERN and MOIST, determine their values from a table which was created during the emulator model initialization. The table contains internal gain as a function of time-of-day. Interpolation is not used with the table and therefore the internal gain is not a continous function of time. Internal gains remain at a constant level until changed by the next entry in the table at a particular time. The internal gain schedule table is stored in five FORTRAN arrays. The arrays contain time-of-day, sensible heat gain from equipment at the corresponding time-of-day, sensible heat gain from people, moisture gain from equipment and moisture gain from people. A function SCHED is used to determine a sensible or moisture gain from the internal gain tables. If moisture gain is requested (by MOIST), SCHED returns the sum of the moisture gain from people and equipment.

If a sensible gain is requested (by INTERN), SCHED returns the sum of the sensible gains from people and equipment.

5.4.5.2 Determining Values for the Internal Gain Schedule Table

The values in the internal gain schedule table are created during the initialization process of the emulator model. An example input parameter file was given in table 2-7. The following equations show how a row in the schedule table is created from the information in a line of table 2-7.

5.4.5.2.1 Convective Heat Gain from People

The internal gain from a person is assumed to depend only on the activity in which the person is engaged and not on the surrounding temperature. Table 5-3 is used as the basis for the sensible and moisture gains from people at various activity levels. The number of occupants at each activity level at a given time is part of the information entered in the building use parameters in table 2-7. The sensible heat generated by the occupants of a zone at a particular time is then assumed to be described by the following equation [Building Environment, NBS (1977), p. A-21]:

$$Q_{people,c} = (1 - r_p) \sum_{k=1}^{N_{act}} n_{p,k} W_{p,s,k} \tag{5-18}$$

where r_p : ratio of radiation energy to total sensible heat of people (=0.4)

$n_{p,k}$: number of people who are doing the same (k-th) activity

$W_{p,s,k}$: sensible heat generation by a person doing the k-th activity.

N_{act} : number of activities

Table 5-3. Occupant sensible and moisture gains

[Table 18 entitled 'Rates of Heat Gain from Occupants of Conditioned Spaces' ASHRAE fundamentals (1981), p.26.25 and Threlkeld, page 363]

	sensible gain (Btu / hr.)	moisture gain (lb. water / hr.)
1. seated, very light work	230.	0.20
2. seated, light work (typing)	255.	0.27
3. standing, light work	315.	0.34
4. light bench work	345.	0.46
5. walking, 3 mph, light machine work	345.	0.74

5.4.5.2.2 Moisture Gain from People

$$M_{people,lat} = \sum_{k=1}^{N_{act}} n_{p,k} W_{p,lat,k} \tag{5-19}$$

where N_{act} : number of activity
$n_{p,k}$: number of people who are doing the same (k-th) activity
$W_{p,lat,k}$: moisture generation by a person doing the k-th activity

5.4.5.2.3 Convective Heat Gain from Equipment and Lighting

The total sensible internal gain from equipment and lighting is determined from the gains due to lighting added to the gain due to equipment. Parameters for the lighting power, lighting type, and equipment sensible gain at various times of the day are entered in the building use parameter file (see table 2-7, for example). For building equipment, the sensible internal gain is described by the following equation [ASHRAE (1981), pp. 26.26 - 26.29]:

$$Q_{equip,c} = (1 - r_e) W_{e,s} \tag{5-20}$$

where r_e : ratio of radiation energy to total equipment energy (=0.2 to 0.8)
$W_{e,s}$: total sensible energy produced by equipment

The sensible internal gain from lighting is assumed to be described by the following equation:

$$Q_{light,c} = (1 - r_\ell)W_\ell \tag{5-21}$$

where W_ℓ : total electric input
r_ℓ : ratio of radiation energy to total light energy
= 0.8 for incandescent lights [Building Environment, NBS (1977),p. A-25]
= 0.5 for fluorescent lights

5.4.5.2.4 Moisture Gain from Equipment

Moisture gain from equipment is specified in the parameters for the building use as shown in table 2-7. The equation for equipment moisture gain is: [ASHRAE (1981), p. 26.26 - 26.28]

$$M_{equip,lat} = W_{e,lat} \tag{5-22}$$

where $W_{e,lat}$ is the total moisture produced by equipment.

5.4.6 Thermal Mass of Zone Air and Furnishings (MASSTH)

The thermal mass which stores energy and dampens fluctuations in zone air temperature is contained in the air in the zone and the zone furnishings, such as desks, cabinets, and partitions. The shell parameter file, an example of which is given in table 2-8, contains the volume of the zone and the thermal capacity of the furnishings. The capacity of the zone air is given by:

$$C_{air} = C_{p,i} \rho_i V_i \tag{5-23}$$

where $C_{p,i}$: specific heat of moist zone air
ρ_i : density of zone air
V_i : volume of zone air (volume of interior space) (parameter, table 2-8)

The total thermal mass of the zone is then:

$$F_{cz} = C_{fur} + C_{air} \tag{5-24}$$

where C_{fur} : thermal mass of furnishings (parameter, table 2-8)

5.4.7 Moisture Gain/Loss due to Infiltration (INFILW)

Moisture gain/loss due to infiltration is assumed to equal a coefficient multiplied by the difference in humidity ratio between zone and outdoor air. The coefficient is determined from:

$$F_{Im} = \rho_{infl} V_i I_{air} \tag{5-25}$$

where ρ_{infl} : density of infiltrated air
I_{air} : air exchange rate (see equation 5-15)
V_i : volume of zone (parameter, table 2-8)

The air exchange rate is determined by the same equation as in section 5.4.2. The density of the infiltration air is determined by taking the average of the densities of the zone air and the outdoor air. This function is considered to be time-dependent.

5.4.8 Moisture Gain/Loss from Supply Air

The moisture added to or removed from the zone by the air handling unit is determined from the difference in the humidity ratios of return and supply air. The equation used is:

$$F_{sw} = W_{sa1}\, m_{sa1} + W_{sa2}\, m_{sa2} - W_{ra}\, m_{ra} \tag{5-26}$$

where: W_{sa1} : humidity ratio of supply air
m_{sa1} : mass flow rate of supply air
W_{sa2} : humidity ratio of hot deck supply air (optional)
m_{sa2} : mass flow rate of hot deck supply air (optional)
W_{ra} : humidity ratio of return air
m_{ra} : mass flow rate of return air

This function is considered to be time-dependent.

5.4.9 Mass of Zone Air and Furnishings that can Absorb Moisture (MASSA)

The mass in the zone which stores moisture and dampens fluctuations in zone humidity ratio is contained in the air in the zone and the zone furnishings, such as curtains, paper, and wood furnishings. The shell parameter file, an example of which is given in table 2-8, contains the volume of the zone and a factor which is used to produce an effective air mass for moisture absorption by the combination of the air and the zone furnishings. This factor is difficult to measure since the moisture absorption properties of office materials are not only difficult to determine but depend on air flow and existing moisture content. This function is considered time-independent. The moisture absorbing capacity of the zone air and furnishings is described by the following equation:

$$F_{mz} = \rho_i\, V_i\, R_{mg} \tag{5-27}$$

where ρ_i : density of zone air
V_i : volume of zone air (volume of interior space)
R_{mg} : a factor used to produce the effective air mass for moisture absorption by the zone furnishings and air.

5.4.10 Overall Thermal Conductance between Wall and Outdoor Air (CNDCTO)

Heat transfer between the mass center of the wall and the outdoor air is a combination of conduction heat transfer within the wall and convection heat transfer from the wall outer surface to the outdoor air. The overall conductance is expressed by:

$$F_{co} = \frac{1}{\dfrac{1}{h_o} + \dfrac{L_{ow}}{k_{ow}}}\, A_w \tag{5-28}$$

where L_{ow} : thickness of wall between the mass center and the outer surface.
k_{ow} : overall conductivity of the wall between the mass center and the outer surface.
h_o : combined heat transfer coefficient (radiative + convective) for heat transfer between the outer wall surface and the outside air.
A_w : surface area of the exterior wall.

The four parameters in this equation are obtained from the shell parameter file, exemplified by table 2-8. This is considered a time independent function.

5.4.11 Thermal Conductance between the Wall and Indoor Air (CNDCTI)

Heat transfer between the mass center of the wall and the inner surface of the wall is by conduction. The thermal conductance is expressed by:

$$F_{ci} = \frac{k_{iw}}{L_{iw}} A_w \qquad (5\text{-}29)$$

where L_{iw} : thickness of wall between the mass center and the inner surface.
k_{iw} : overall conductivity of the wall between the mass center and the inner surface.
A_w : surface area of the exterior wall.

The three parameters in this equation are obtained from the shell parameter file, exemplified by table 2-8. This is considered a time independent function.

5.4.12 Thermal Capacitance of the Wall

The mass in the exterior wall which stores energy and dampens fluctuations in wall temperature is contained in the building materials of which the wall is constructed. The wall is divided into an inner and outer section by the mass center of the wall. The shell parameter file, an example of which is in table 2-8, contains the thickness, density, thermal conductivity and specific heat of the inner and outer wall sections. This function is considered time-independent. The thermal capacitance of the exterior wall is described by the following equation:

$$F_{cw} = (C_{p,ow}\, \rho_{ow}\, L_{ow} + C_{p,iw}\, \rho_{iw}\, L_{iw})\; A_w \qquad (5\text{-}30)$$

where $C_{p,ow}$, $C_{p,iw}$: average specific heat of outer and inner sections of wall.

ρ_{ow}, ρ_{iw} : average density of outer and inner sections of wall.

L_{ow}, L_{iw} : thickness of outer and inner sections of wall.

A_w : area of the exterior wall.

5.4.13 Radiative Heat Transfer Coefficient (WRADIA)

Radiative heat transfer between the inner surface of the exterior wall and the other surfaces in the zone is assumed to be equal to a radiative heat transfer coefficient multiplied by the difference in the mean radiant temperature of the zone and the wall surface temperature. For simplicity, this mode of heat transfer was not implemented in the abridged building emulator model program. Determination of this mode of heat transfer involves the determination of view angle factors of surfaces which enclose the zone. [Walton (1980), Sowell and Walton (1980)]. Using the view angle, the radiant heat transfer coefficient could then be determined [Walton (1980), p. 69]. However, it is necessary to know the emissivities and the surface temperatures of all zone surfaces, as well as the specific geometry. The abridged model effectively assumes that the radiative coefficient is small or the difference between the surface and mean radiant temperatures is negligible.

5.4.14 Solar Energy Entering the Zone Absorbed by Walls (WSOLAR)

If the zone contains a glass window, solar energy may enter the zone through the window and be absorbed by the walls and furnishings. For simplicity, this mode of heat transfer was not implemented in the abridged building emulator model program. Since the only zone wall which has inner surface temperature as a state variable is the exterior wall, and this is the wall which is assumed to have a glass window, no solar energy from the window would be absorbed by the exterior wall. Solar energy could be absorbed by the furnishings, however. The abridged model effectively assumes that the zone walls are without windows. The equation for solar energy absorption is included in this section for completeness. If it is assumed that solar irradiation is not absorbed by the glass window:

$$F_{ts} = \frac{A_g \, r_g \, S_c \, I}{A_w} \qquad (5\text{-}31)$$

where A_g : glass window area

r_g : solar transmittance through glass [ASHRAE (1981), p. 27.38]

I : total solar radiation on the glass window

S_c : shading coefficient [ASHRAE (1981), p.27.38]

5.4.15 Thermal Conductance Through Glass to Outdoor Air (GCNDCT)

Heat transfer between the inner surface of the glass window and the outdoor air is a combination of conduction heat transfer within the glass and convection heat transfer from the glass outer surface to the outdoor air. This mode of heat transfer is not included in the abridged emulator model program since the abridged model assumes that there are no windows in the zone. The overall conductance equation is included for completeness and is expressed by:

$$F_{Ag} = \frac{1}{\dfrac{1}{h_o} + \dfrac{L_g}{k_g}} A_g \qquad (5\text{-}32)$$

where L_g : thickness of glass.
k_g : conductivity of the glass.
h_o : combined heat transfer coefficient (radiative + convective) for heat transfer between the outer glass surface and the outside air.
A_g : surface area of glass window.

5.4.16 Thermal Convection from Inner Glass Surface to Indoor Air (GCNVCT)

Heat transfer between the inner glass surface and the zone air is assumed to be equal to a convective heat transfer coefficient multiplied by the temperature difference between the inner glass surface and the zone air. The heat transfer coefficient must be supplied in the building shell parameter file. This mode of heat transfer is not included in the abridged version of the building emulator model program since the abridged model assumes that there are no windows in the zone.

5.4.17 Radiant Heat Transfer between Glass and Zone (GRADIA)

Radiative heat transfer between the inner surfaces of windows and the other surfaces in the zone is assumed to be equal to a radiative heat transfer coefficient multiplied by the difference in the mean radiant temperature of the zone and the glass surface temperature. Since there are assumed to be no windows in the zone, this mode of heat transfer was not implemented in the abridged building emulator model program. Determination of this heat transfer is equivalent to the process described in section 5.4.13.

5.4.18 Mean Radiation Temperature (MRT)

The mean radiant temperature is used to determine radiation heat transfer between the zone environment and wall or glass surfaces. The mean radiant temperature is a function of wall and glass surface temperatures, the radiant heat from people, the radiant heat from equipment, the properties of the surfaces, and the geometry of surfaces [Walton (1983), p. 69]. Radiation heat transfer is not included in the abridged version of the emulator model program and the equation for mean radiant temperature is not included here.

5.5 Property Equations for Moist Air

The building emulator model program uses a number of functions to determine properties of moist air. The equations used are not presented in this report, but the names of the computer routines, the property determined, and a reference for the equations, is given in table 5-4.

5.6 Local Equipment Model

The local heating and cooling equipment is defined as the equipment located in or adjacent to the zone which is designed to directly heat or cool only that zone. The purpose of this equipment is to adjust the energy in the incoming air from the air handling unit to match the heating/cooling requirements of the zone. There are three different types of local equipment which may be present. These are: 1. energy consuming local equipment such as reheat coils, fan coils, heat pumps, perimeter radiation, or electric heaters; 2. air volume reduction controls to adjust the volume of supply air to match the zone requirements for VAV systems; and 3. air mixing controls to mix two air streams of different temperatures to match the zone requirements for multizone or dual-duct systems. Local equipment is usually under local control, which means that the controller for the equipment is located adjacent to the zone and the control sensor is located in the zone.

The subroutine LOCAL in the building emulator model program is used to simulate local equipment and determine energy usage required to maintain the zone air temperature as close to a local setpoint as possible. This subroutine is called by the ZONE subroutine for each zone model time step in the solution of the zone model equations.

Table 5-4 Moist air property routines

CPAIR	: specific heat of moist air as a function of humidity ratio [ASHRAE fundamentals 1981, page 5.3]
RHOAIR	: density of moist air as a function of humidity ratio and dry bulb temperature [ASHRAE fundamentals 1981, page 5.3]
ENTHAL	: enthalpy of moist air as a function of dry bulb temperature, humidity ratio, and atmospheric pressure [ASHRAE fundamentals 1981, page 5.4]
TEMP	: dry bulb temperature of moist air as a function of enthalpy, humidity ratio, and atmospheric pressure [ASHRAE fundamentals 1981, page 5.4]
DEWPT	: dew point temperature of moist air as a function of humidity ratio, and atmospheric pressure [ASHRAE fundamentals 1981, page 5.4, equation 40b]
HUMRAT	: Humidity ratio of moist air as a function of dry bulb temperature, relative humidity, and atmospheric pressure [ASHRAE fundamentals 1981, page 5.4]
RELHUM	: Relative humidity of moist air as a function of dry bulb temperature, humidity ratio, and atmospheric pressure [ASHRAE fundamentals 1981, page 5.4]
PWSAT	: Saturation pressure of water vapor as a function of absolute temperature [ASHRAE fundamentals 1981, page 5.2]
TDBSAT	: dry bulb temperature of saturated air as a function of the enthalpy of saturated air and the atmospheric pressure [ASHRAE fundamentals 1981, page 6.5, table 2]

The zone model will require a value for the energy being added to or removed from the zone due to the HVAC system. This energy, designated as q_{ent}, which enters or leaves the zone, consists of two parts, the energy removed or added by the air from the air handling unit, q_{air}, and the energy removed or added by local energy consuming equipment, q_{local}. The q_{local} energy may not be present on certain systems. q_{air} can be calculated by:

$$q_{air} = m_{sa1}\, h_{sa1} + m_{sa2}\, h_{sa2} - h_i\, m_{ra} \tag{5-33}$$

where m_{sa} are supply air mass flow rates (1 indicates the cold deck and 2 the optional hot deck for a dual deck system), h_{sa} are the supply air enthalpies,

m_{ra} is the return air mass flow rate, and h_i is the interior space enthalpy of the zone.

The actions that the local equipment model simulates are those of the local controller which will control the local equipment to maintain a zone air temperature at a setpoint. If the actual zone air temperature deviates from the setpoint, the local controller will attempt to return the temperature to the setpoint by increasing or decreasing the local equipment output. A simple proportional plus integral control (PI-control) model will be used. For this model, the adjustment the local controller attempts to make to q_{ent} is determined from:

$$dq_{ent} = K_p\ (e-e_0) + K_I\ (e\ dt) \tag{5-34}$$

where e, error = $T_{setpoint} - T_i$, e_0 is the previous value of the error, K_I is an integral gain factor, K_p is a proportional gain factor, and dt is the zone model time step. Once dq_{ent} is determined, the type of HVAC system determines how q_{air} and q_{local} are to be determined.

If the system is dual deck, it is assumed that there is no local conditioning equipment beyond the mixing box for the two deck air streams. Then:

$$dq_{air} = dq_{ent}, \tag{5-35}$$

$$q_{air(t+dt)} = q_{air(t)} + dq_{air} \tag{5-36}$$

The total mass flow rate of air from the air handling unit is assumed to be constant at m_{sa}. The supply air enthalpy after mixing must then be:

$$h_{sa} = \frac{q_{air}}{m_{sa}} + h_i \tag{5-37}$$

Since the flow rates of the two decks must sum to equal the total mass flow rate, and the two deck air streams are assumed to mix ideally and have a known supply air temperature, humidity ratio, and enthalpy, the unknown mass flow rates for the decks can be solved for:

$$m_{sa2} = \frac{m_{sa}\ (\ h_{sa} - h_{sa1}\ )}{h_{sa2} - h_{sa1}} \tag{5-38}$$

$$m_{sa1} = m_{sa} - m_{sa2} \tag{5-39}$$

also: $$W_{sa} = m_{sa1}\ W_{sa1} + m_{sa2}\ W_{sa2} \tag{5-40}$$

If the system is a single deck VAV, it is assumed that there is no local conditioning equipment beyond the VAV box for modulating the supply air flow rate into the zone. This may not be true for some systems, for example VAV reheat systems. Then:

$$dq_{air} = dq_{ent}, \tag{5-41}$$

$$q_{air(t+dt)} = q_{air(t)} + dq_{air} \tag{5-42}$$

The enthalpy of the air from the air handling unit is assumed to be constant at h_{sa}. The mass flow rate, m_{sa}, can be varied. The supply air flow rate for the new value of energy addition to the zone must then be:

$$m_{sa} = \frac{q_{air}}{h_{sa} - h_i} \tag{5-43}$$

The local zone VAV box will have a maximum and minimum air flow rating. If these limits are exceeded, the air flow rate will be constrained to the limits. If this is the case, the energy entering the room will have to be recalculated based on the limited mass flow rate.

If the system is single deck constant volume, it is likely to have local conditioning equipment. In this case:

$$dq_{local} = dq_{ent}, \tag{5-44}$$

$$q_{local(t+dt)} = q_{local(t)} + dq_{local} \tag{5-45}$$

If q_{local} exceeds the local capacity, it will remain at the local capacity. If q_{local} becomes negative, this represents a need for local cooling. If the local equipment has no cooling capacity, such as a reheat coil, then q_{local} will be set to zero. If cooling capacity is present, then q_{local} can take on negative values up to the cooling capacity of the local equipment.

6. OCCUPANT COMFORT MODEL

6.1 Assumptions and Purpose of Comfort Model

The purpose of the occupant comfort model is to determine in a basic way if the conditions in the zone are in accordance with ASHRAE standard 55-74 for thermal comfort. This standard basically assumes that comfort is acceptable if the zone dry bulb temperature and humidity ratio are within certain limits.

The test for comfort is only performed if the building zone is determined to be in an occupied state. The zone is assumed to be occupied if the internal gain schedule table, contained in the model function SCHED, predicts that the sensible gain from people is non-zero for the current time-of-day. (see section 5.4.5.

6.2 Comfort Envelope Equations

The equations to determine if the current state of zone air is within the ASHRAE comfort limits are based on the simple two-dimensional geometry of the psychometric chart. The upper and lower comfort limits for humidity ratio are constant and this allows a simple comparison of zone humidity ratio to the limits. If the humidity ratio in the zone is greater than 0.012 or less than 0.0044 pounds of water per pound of dry air then the zone is automatically outside of comfort limits.

The maximum and minimum comfort limits for dry bulb temperature are a linear function of humidity ratio between the upper and lower humidity ratio limits as expressed by:

$$T_{\text{lower limit}} = 72.58 - 131.58\, W_{za}, \quad \text{and} \tag{6-1}$$

$$T_{\text{upper limit}} = 81.73 - 394.74\, W_{za}. \tag{6-2}$$

where W_{za} is the zone humidity ratio in pounds of water per pound of dry air, and T is the dry bulb temperature in degrees Fahrenheit. If the zone dry bulb temperature is above or below these limits, then the zone is outside of comfort limits.

The total time that the zone is outside of comfort limits during occupied hours is determined by the comfort model subroutine. The maximum and minimum values of zone dry bulb temperature and relative humidity are also recorded.

7. COMPILATION OF ENERGY USE

The predicted energy used by the emulated building during a test run of the emulator is determined by the FORTRAN subroutine CMPILE. CMPILE uses the values of the temperature, enthalpy, and air flow rate variables in the state vector to calculate the energy used between the current and previous time steps in several categories, and essentially integrates the energy use by keeping cumulative totals which are placed in the state vector.

7.1 Categories of Energy Use

Energy used by a building is assumed to be categorized as follows:

1. energy used by the fans in the air handling unit (the power used by the fans will be constant unless a variable air volume system is used, in which case fan power will vary as a function of air flow).

2. energy used to heat or cool the air passing through the air handling unit (this would not include cooling by outside air).

3. energy used by local space heating or cooling equipment (reheat coils, heat pumps, fan coils, perimeter radiation, air conditioners).

It is assumed that the calculated energy values are not in terms of any fuel and do not include any plant efficiencies. The energies are either in energy added to the air and extracted from steam or hot water for heating, or in energy required to be mechanically extracted from the air by chilled water or refrigerant for cooling.

There are 4 primary sources of reduction in energy use which EMCS algorithms may cause. Values for the energy reductions may be calculated from the above types of energy use in a building. These are:

1. savings from not running electrical loads (fans, pumps).
2. savings from reducing the daily heating or cooling requirements of a building by reduction of operating hours or changing of local space temperature setpoints.
3. savings by using the outside air for heating or cooling and avoiding mechanical heating or cooling.
4. savings from reduction of reheating or recooling of air passing though the air handler and building space for purposes of humidity control and compensation for load variation between zones. These causes of energy consumption are sometimes termed system effects, since the amount of energy for this purpose will vary between different HVAC systems, such as between reheat and multizone.

7.2 Equations for Energy Use

The first category of energy consumption is air handling unit fan energy consumption. If the system is constant volume, the steady-state power consumption of the fan can be used to calculate energy. If the system is variable air volume, it is assumed that the fan power consumption will vary with the volume flow rate through the unit. If the rated power consumption of a fan is known at a rated flow rate, then the power consumption at another flow rate can be determined based on fan law number 3 from the ASHRAE equipment handbook [1983, page 3.5, table 2]:

$$P_{fan} = P_{rated} \left(V_a / V_{rated} \right)^3 \tag{7-1}$$

where: P_{rated} : power consumption of fan at rated air flow rate.
V_{rated} : rated fan volumetric air flow rate.
V_a : current volumetric flow rate.

The fan energy between the current and last time step is determined by multiplying P_{fan} by the emulator model time step. The cumulative energy for the fan is determined by adding the current time step energy to the sum of values of energy for previous time steps.

The thermal energy used for heating and cooling the building should be divided into the energy added to the building air for heating purposes and the energy removed from the building air for cooling purposes. Therefore the energy values will always have two parts, heating and cooling. The summation or subtraction of two energies will then consist of the separate summation or subtraction of the heating and cooling parts (this is analogous to two-dimensional vectors with x and y components). For the following equations, an energy rate is calculated. This rate is multiplied by the emulator model time step to determine the energy used since the last time step. The cumulative energy is obtained by adding this to the sum of energy values of previous time steps.

The requirements energy is the energy required to bring the air leaving the building zones through the return ducts and the ventilation air to the level of the supply air enthalpy (including both air streams if dual deck).

The requirements energy rate (power) is:

$$q_{req} = m_{sal} \left(h_{sal} - h_{mc} \right) \tag{7-2}$$

where the mc subscript indicates the mixed air temperature in the air handling unit with only minimum outside air or ventilation air (see section 4.2). If $q_{req} < 0$ this is cooling energy, if $q_{req} > 0$ this is heating energy. If the system is dual deck there is an additional component of the requirements energy rate for the hot deck.

$$q_{req2} = m_{sa2} (h_{sa2} - h_{mc}) \tag{7-3}$$

If $q_{req2} < 0$ this is cooling energy, if $q_{req2} > 0$ this is heating energy. The hot deck component is added to the cold deck component of requirements energy, keeping heating and cooling energies separated.

The actual energy is the energy required to bring the air passing through the air handler to the level of the supply air enthalpy from the level of the mixed air enthalpy. This is usually equal to the energy supplied or removed by the heating and cooling coils. This actual energy will be less than the requirements energy since part of the requirements energy can be met by an economizer algorithm using outdoor air for cooling.

The actual energy rate (q_{act1} for single or cold deck and q_{act2} for hot deck if a dual deck system) is then:

$$q_{act1} = m_{sa1} (h_{sa1} - h_{ma}) \tag{7-4}$$

$$q_{act2} = m_{sa2} (h_{sa2} - h_{ma}) \tag{7-5}$$

q_{act1} and q_{act2} are summed, keeping heating and cooling energies separated.

The requirements and actual energy can be used to determine the economizer energy. This is not usually considered an energy use, since the outdoor air costs nothing to condition. However, for purposes of calculating energy savings for economizer algorithms (savings category 3 above) it is useful to consider this as energy consumption. The economizer energy rate is:

$$q_{econ} = q_{req} - q_{act} \tag{7-6}$$

The actual energy can be divided into two parts which are the other two categories of building energy use. The energy that would be required to bring the mixed air to the actual condition of the air entering the building space (after mixing for a dual deck system) can be termed the load energy. This will equal the actual energy for all systems except dual deck. This represents the energy requirements with system effects minimized (used for savings in category 2, above). The load energy is:

$$q_{load} = (m_{sa1} h_{sa1} + m_{sa2} h_{sa2}) - (m_{sa1} + m_{sa2}) h_{ma} \tag{7-7}$$

The final category of energy use is the energy used for reheating and recooling of air (used for savings in category 4, above). This is defined as:

$$q_{rhc} = q_{act} - q_{load} + q_{local} \tag{7-8}$$

This sum is of the heating and cooling components of the energies.

When a test run using the emulator is complete, the emulator model program will generate a summary report file containing energy usage for the emulated building over the test period. This task is performed by subroutine REPORT. The report will contain energy usage in the categories discussed above for heating and cooling energy components. An example energy report is given in table 7-1. The report file will also contain results on the amount of time the zone was not within comfort limits and the maximum and minimum space temperatures and relative humidities reached during occupied hours.

Table 7-1. Heating and cooling energy from emulator model

FAN ENERGY		
	HEATING	COOLING
ECONOMIZER ENERGY		
LOAD ENERGY		
REHEAT/RECOOL ENERGY		

8. SUMMARY

The use of a building emulator is one of the approaches to testing of EMCS systems without making actual connections to HVAC equipment or a building. The NCEL/NBS emulator will be used for either factory or field acceptance tests of EMCS algorithms. The building simulation program in the emulator is the main part of this report. This program simulates the responses of a building to EMCS control actions.

The building emulator program was written in FORTRAN 77 and the source code is appended in the APPENDIX. State variables in the emulator model are analog sensor, energy, command, and comfort variables. The building emulator model program contains the weather (WEATHR), air handling unit (AHU), zone (ZONE), and comfort (COMFRT) model. The use of energy during a simulation is accumulated by the energy compilation subroutine (COMPILE). This energy use information is useful to evaluate the performance of an EMCS algorithm.

Prior to simulation, data files of initial values of state variables, building shell parameters, HVAC system parameters, and weather data parameters must be called. The air handling unit model is a steady-state model, while the zone model is a dynamic model. In the zone model, a system of first order ordinary differential equations is solved by the explicit forward difference scheme. The time step for the difference scheme is adjusted according to the changes of state variables. By using this adaptive time step, which is uniformly distributed in a given emulator simulation time period, savings in computation time can be realized when the state variables approach steady state.

The air handling unit model simulates a single deck (or single duct), constant volume system without humidity control as shown in Figure 4-2. Damper positions of the air handling unit are varied if economizer option is selected. A single zone with exterior walls without window is modeled under the assumptions described previously. For this abridged system, a local equipment model is implemented. Heating/cooling energy delivered from the local equipment to zone air is adjusted under the control law of the proportional plus integral controller (PI-controller). Capabilities of simplified current models can be upgraded by providing necessary information in the spaces already reserved for future extension.

9. REFERENCES

ASHRAE, <u>1981 Fundamentals Handbook, ASHRAE, 1981</u>.

Berglund, L., 'Control and Simulation of Thermal Comfort,' Workshop on HVAC Control, Modeling, and Simulation, Atlanta, GA, Feb. 2-3, 1984.

Borresen, B.A., 'Thermal Room Models for Control Analysis,' ASHRAE Trans., Vol. 87, Part 2, 1981.

Brokaw, R.S., 'Calculation of Flue Losses for High-Efficiency Furnaces and Appliances,' ASHRAE Journal, Jan. 1979, pp. 49-51.

Building Environment Division of NBS, 'Technical Guidelines for Energy Conservation,' AFCEC-TR-77-12 and NBSIR 77-1238, Nat'l Bureau of Standards, June 1977.

Bullock, C.E., 'Dynamic Simulation Models for Commercial Air Conditioning and Heat Pump Systems,' Workshop on HVAC Control, Modeling, and Simulation, Atlanta, GA, Feb. 2-3, 1984.

Chi, J., 'Building Environmental Control System Analysis Computer Program - Volume 1 Component Mathematical Models,' Final Report submitted to NBS, September 1982.

Fanger, P.O., <u>Thermal Comfort</u>, McGraw-Hill, 1970.

Ferziger, J.H., <u>Numerical Methods for Engineering Application</u>, John Wiley, 1981.

Kao, J.Y., 'Strategies for Energy Conservation for a Large Office Building,' NBSIR 83-2746, Nat'l Bureau of Standards, July 1983.

Kusuda, T., 'NBSLD, The Computer Program for Heating and Cooling Loads in Buildings,' NBS BSS 69, Nat'l Bureau of Standards, 1978.

May, W.B., 'Time of Day Control and Duty Cycling Algorithms for Building Management and Control Systems,' NBSIR 83-2713, Nat'l Bureau of Standards, June 1983.

May, W.B., 'Control Algorithms for Building Management and Control Systems--Hot Deck/Cold Deck/Supply Air Reset, Day/Night Setback, Ventilation Purging, and Hot and Chilled Water Reset,' NBSIR 83-2713, Nat'l Bureau of Standards, March 1984.

Modest, M.F. and Macken, N., 'Seasonal Performance Comparisons for an Apartment Building in New York State Climate Regions,' ASHRAE Trans., 1977, pp. 881-892.

Park, C. 'Single-zone Computer Model for Residential Furnace Location Analysis,' ASHRAE Trans., Vol. 87, Part 2, 1981.

Park, C., Kelly, G.E. and Kao, J.Y. 'Economizer Algorithms for Energy Management and Control Systems,' NBSIR 84-2832, Nat'l Bureau of Standards, Feb. 1984.

Park, C., 'An Optimum Start/Stop Control Algorithm for Heating and Cooling Systems in Buildings,' NBSIR 83-2720, Nat'l Bureau of Standards, May 1983.

Park, C., 'Demand Limiting Algorithms for Energy Management and Control Systems,' NBSIR 84-2826, Nat'l Bureau of Standards, February 1984.

Sowell, E.F. and Walton, G.N., 'Efficient Computation of Zone Loads,' ASHRAE Trans., Vol. 86, Part 1, 1980.

Threlkeld, J.L., Thermal Environmental Engineering, 2nd ed., Prentice-Hall, 1970.

Walton, G.N., 'A New Algorithm for Radiant Interchange in Room Loads Calculations,' ASHRAE Trans., Vol. 86, Part 2, 1980.

Walton, G.N., 'Thermal Analysis Research Program Reference Manual,' NBSIR 83-2655, Nat'l Bureau of Standards, March 1983.

Wise, B.B., 'Energy Monitoring and Control System Software Testing Device,' Ninth Energy Management & Controls Society Conference Proceedings, Nov. 1984.

APPENDIX

COMPUTER PROGRAM LISTINGS

```
C Version 1.7 - FEBRUARY 05, 1985 - W.B. MAY ,NATIONAL BUREAU OF STANDARDS
C=========================================================================
      PROGRAM ALGTST
C=========================================================================
C This is the main program used to develop and test simulation routines
C for a simple building model to be used in an EMCS algorithm tester
C developed by the U.S. Navy Civil Engineering Laboratory. This test
C program takes the place of the EMCS tester software. Subroutine
C EMODEL is the emulator model. Subroutine UPDATE represents the I/O
C task which determines if any commands have arrived from the EMCS and
C updates the analog and digital signals to the EMCS based on the
C simulation calculations. It is assumed that these routines will
C eventually run in parallel, UPDATE at a higher frequency. Parallel
C execution is approximated by serial calls, but with UPDATE being
C called more often.
C
C Determine start time
      CALL SYNCCL
C-------------------------------------------------------------------------
C Update commands and outputs
C
 1000 CALL UPDATE
C-------------------------------------------------------------------------
C Call the emulator model
C
      CALL EMODEL
C-------------------------------------------------------------------------
C Update clock
C
      CALL ECLOCK
      GO TO 1000
      END
C=========================================================================
      BLOCK DATA COMDAT
C=========================================================================
C This module contains a list of all common block data and globals which
C are referenced in the support routines for the EMODEL subroutine.
C Additional common blocks used only with EMODEL are described in the
C subroutine EMODEL or the subroutine PARAMA.
C   common blocks are:
C ASENSR - contains all analog sensor readings in arrays. Sensor
C          readings are copied from the emulator model outputs after
C          each time step. Array index indicates which of 5 possible
C          buildings the reading is from.
C COMMND - contains all command digital inputs from the EMCS in arrays.
C          readings are directly available to the emulator model.
C CPA    - contains all control point adjustments from the EMCS in arrays.
C DSENSR - contains all digital sensor readings in arrays.
C EVENT  - GLOBAL which is a logical variable indicating an EMCS command.
C PAST -   contains day, hour, minute, and second that model subroutine
C PFILES - contains the names of all parameter files,and the season.
C          was last called, and the simulation major timestep.
C STATE1 - contains the state variables for the emulator model.
C TIME -   current time is broken into DAY, HOUR, MINUTE, SECOND. In this
C          test program, all times are zero at the start of the program.
C          the basic time step of the program is one second. ENDDAY,ENDHR,
C          ENDMIN, and ENDSEC are the scheduled time to end the simulation.
```

```
C UNITS -  contains the logical units used by the program for various
C          inputs and outputs.
C---------------------------------------------------------------------
C
C TOUTA  - EMCS analog input, Temperature, Outside Air (F)
C RHOUTA - EMCS analog input, Relative Humidity, Outside Air (%)
C TZONEA - EMCS analog input, Temperature, Space  (F)
C RHZONA - EMCS analog input, Relative Humidity, Space  (%)
C TRETA  - EMCS analog input, Temperature, Return Air  (F)
C RHRETA - EMCS analog input, Relative Humidity, Return Air (%)
C ZHCDEM - EMCS analog input, Position, Reheat Valve (Btu/s)
C TSUPA  - EMCS analog input, Temperature, Supply Air  (F)
C TMIXA  - EMCS analog input, Temperature, Mixed Air  (F)
C
      REAL                TOUTA,RHOUTA,TZONEA,   RHZONA,   TRETA,
     &                    RHRETA,   ZHCDEM,   TSUPA   ,TMIXA
      COMMON / ASENSR / TOUTA,RHOUTA,TZONEA(5),RHZONA(5),TRETA(5),
     &                    RHRETA(5),ZHCDEM(5),TSUPA(5),TMIXA(5)
      DATA TOUTA,RHOUTA,TZONEA,RHZONA,TRETA,RHRETA,ZHCDEM/27*0.0/
      DATA TSUPA,TMIXA/10*0.0/
C
C ON     - EMCS command, Supply Fan On/Off
C ECON   - EMCS command, O.A. Damper Off/Auto
C VENT   - EMCS command, Ventilation Damper Open/Close
C SETBAK - EMCS command, Zone Thermostat Setback On/Off
C
      LOGICAL             ON,   ECON,   VENT,   SETBAK
      COMMON / COMMND / ON(5),ECON(5),VENT(5),SETBAK(5)
      DATA ON,ECON,VENT,SETBAK/20*.FALSE./
C
C SUPPLY - EMCS CPA, Supply Air Temperature (F)
C ZONE     EMCS CPA, Zone Temperature (F)
C
      REAL                SUPPLY   ,ZONE
      COMMON /  CPA   / SUPPLY(5),ZONE(5)
      DATA SUPPLY,ZONE/10*0.0/
C
C DPSSF  - EMCS digital input, differential Pressure Switch, Supply Fan
C
      LOGICAL             DPSSF
      COMMON / DSENSR / DPSSF(5)
      DATA DPSSF/5*.FALSE./
C
C EVENT  - Logical variable which is true when a command from the EMCS
C          file has been received.
C
      LOGICAL EVENT
      GLOBAL EVENT
C
C OLDDAY - Day number when EMODEL subroutine was last called.
C OLDHR  - Hour of day when EMODEL subroutine was last called.
C OLDMIN - Minute of hour when EMODEL subroutine was last called.
C OLDSEC - Second of minute when EMODEL subroutine was last called.
C STEP   - Major time step for simulation. model routines for AHU ,
C          weather, zone, comfort, and energy compilation are called
C          at this interval. Not the same as the Euler timestep used
C          within the zone model.
C
```

```
      INTEGER            OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
      COMMON / PAST    / OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
      DATA OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP/4*0,30/
C
C HVFILE - Character filename of HVAC equipment parameter file.
C USFILE - Character filename of Building usage parameter file.
C CLFILE - Character filename of climate description parameter file.
C SHFILE - Character filename of building shell parameter file.
C INFILE - Character filename of state vector initialization file.
C SEASON - Character name of weather season to use in test.
C
      CHARACTER*15       HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
      COMMON / PFILES / HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
C
C A      - Real array containing state variables at the current time.
C L      - Logical array containing current logical state variables.
C
      LOGICAL                      L
      REAL               A
      COMMON / STATE1 / A(100),L(100)
      DATA A/100*0.0/,L/100*.FALSE./
C
C DAY    - Current day number, first day of test is day 1.
C HOUR   - Current hour of day, from 0 - 23.
C MINUTE - Current minute within the hour, 0 - 59.
C SECOND - Current second within the minute, 0 - 59.
C ENDDAY - Day number on which the test is to stop.
C ENDHR  - Hour of the day on ENDDAY within which test is to stop.
C ENDMIN - Minute within ENDHR within which test is to stop.
C ENDSEC - Second within ENDMIN within which test is to stop.
C
      INTEGER           DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME   / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
C CLU    - Logical Unit for writes to console.
C FLU    - Logical Unit for read from EMCS command input file.
C ILU    - Logical Unit for read from console.
C PLU    - Logical Unit for read from paramater files.
C
      INTEGER            CLU,FLU,ILU,PLU
      COMMON / UNITS  / CLU,FLU,ILU,PLU
      DATA CLU/1/,FLU/8/,ILU/1/,PLU/9/
C
      END
C======================================================================
      SUBROUTINE SYNCCL
C======================================================================
C This routine is used to determine the start time of the test. In a
C real-time system, this routine would set the real time clock to the
C proper time.
C
C Input variables: none
C Output variables: none
C Procedures called: none
C Common blocks: UNITS,TIME
C
      INTEGER            CLU,FLU,ILU,PLU
      COMMON / UNITS / CLU,FLU,ILU,PLU
```

```
C
      INTEGER          DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
C Local Variables:
C CFFILE - Character name of file containing configuration information.
C
      CHARACTER*15 CFFILE
      PARAMETER (CFFILE = 'BLDG.CNF')
C
C-----------------------------------------------------------------------
C Read the configuration file to determine the start time.
C
      OPEN(PLU,FILE=CFFILE,STATUS='OLD',ERR=9800)
      READ(PLU,*) HOUR,MINUTE,SECOND
      CLOSE(PLU)
      DAY = 1
      RETURN
C--------ERRORS---------------------------------------------------------
 9800 WRITE(1,1)
    1 FORMAT(1X,'Cannot successfully open configuration file')
      STOP
      END
C=======================================================================
      SUBROUTINE CONPAR
C=======================================================================
C This routine is used to determine all of the emulator routine control
C parameters including duration of test, parameter file
C names and weather season.
C
C Input variables: none
C Output variables: none
C Procedures called: none
C Common blocks: UNITS,TIME,PAST,PFILES
C
      INTEGER            CLU,FLU,ILU,PLU
      COMMON / UNITS / CLU,FLU,ILU,PLU
C
      INTEGER          DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
      INTEGER          OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
      COMMON / PAST / OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
C
      CHARACTER*15       HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
      COMMON / PFILES / HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
C
C Internal Variables:
C
C CFFILE - Character name of file containing configuration information.
C
      CHARACTER*15 CFFILE
      PARAMETER (CFFILE = 'BLDG.CNF')
C
C-----------------------------------------------------------------------
C Read the configuration file to determine the test duration, HVAC type
C file, building use file, weather data file, building type file,
C initialization file, season, and simulation time step.
```

```
C
      OPEN(PLU,FILE=CFFILE,STATUS='OLD',ERR=9800)
      READ(PLU,*)
      READ(PLU,*) ENDDAY,ENDHR,ENDMIN,ENDSEC
      READ(PLU,*) HVFILE
      READ(PLU,*) USFILE
      READ(PLU,*) CLFILE
      READ(PLU,*) SHFILE
      READ(PLU,*) INFILE
      READ(PLU,*) SEASON
      READ(PLU,*) STEP
      CLOSE(PLU)
C-----------------------------------------------------------------------
C Compute ending time
C
      ENDDAY = ENDDAY + DAY
      ENDSEC = ENDSEC + SECOND
      IF(ENDSEC.GT.59)THEN
        ENDSEC = 0
        ENDMIN = ENDMIN + 1
      ENDIF
      ENDMIN = ENDMIN + MINUTE
      IF(ENDMIN.GT.59)THEN
        ENDMIN = 0
        ENDHR = ENDHR + 1
      ENDIF
      ENDHR = ENDHR + HOUR
      IF(ENDHR.GT.23)THEN
        ENDHR = 0
        ENDDAY = ENDDAY + 1
      ENDIF
      OLDDAY = DAY
      OLDHR = HOUR
      OLDMIN = MINUTE
      OLDSEC = (-STEP)
      RETURN
C--------ERRORS---------------------------------------------------------
 9800 WRITE(1,1)
    1 FORMAT(1X,'Cannot successfully open configuration file')
      STOP
      END
C=======================================================================
      SUBROUTINE ECTROL(TIMEST,GO)
C=======================================================================
C This routine is used to control the execution timing of the emulator
C model routine. If GO is returned as TRUE, the emulator simulation
C should be executed. This occurs when 1. the time step duration has
C passed, and 2. a control event has been detected. TIMEST contains the
C time step that the model is to use in simulation.
C
C Input variables: none
C
C Output variables:
C TIMEST - The current value to be used for the major simulation time step.
C GO     - A logical variable which indicates that the simulation
C          subroutine should be executed if true.
      LOGICAL GO
      INTEGER TIMEST
```

```
C
C Procedures called: NUSTEP, TIMINC
C
C Common blocks: none
C
C Global Variables:
C EVENT  - Logical variable which is true when a command from the EMCS
C           file has been received.
C
      LOGICAL EVENT
      GLOBAL EVENT
C
C Local Variables:
C STEPDN - Logical variable, true if time for another simulation execution.
C TIMEUP - Logical variable, true if test ending time has been reached
C
      LOGICAL TIMEUP,STEPDN
C
C-----------------------------------------------------------------------
C Update the time and determine if the execution timestep has passed.
C If so, STEPDN will be true.  Also determine if the ending time has
C been reached. If so, TIMEUP will be true.
C
      CALL TIMINC(STEPDN,TIMEUP)
      IF(TIMEUP) THEN
       CALL REPORT
       STOP 'Emulation Complete'
      ENDIF
      IF(EVENT.OR.STEPDN)THEN
        GO = .TRUE.
        STEPDN = .FALSE.
        CALL NUSTEP(TIMEST)
      ELSE
        GO = .FALSE.
      ENDIF
      RETURN
      END
C=======================================================================
      SUBROUTINE NUSTEP(TIMEST)
C=======================================================================
C This routine returns the amount of time in seconds since the last time
C it was called.
C
C Input variables: none
C Output variables:
C TIMEST - Time in seconds since routine was last called.
C
      INTEGER TIMEST
C
C Procedures called: SBTIME
C Common blocks: TIME,PAST
C
      INTEGER DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
      INTEGER OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
      COMMON / PAST / OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
C
```

```
C Local Variables: none
C
C-------------------- determine time since last call ------------------
      CALL SBTIME(DAY,HOUR,MINUTE,SECOND,OLDDAY,OLDHR,OLDMIN,OLDSEC,
     & TIMEST)
C-------------------- reset last time called variables ----------------
      OLDDAY=DAY
      OLDHR=HOUR
      OLDMIN=MINUTE
      OLDSEC=SECOND
      RETURN
      END
C=======================================================================
      SUBROUTINE ECLOCK
C=======================================================================
C This routine increments the time. In a real-time system, the time would
C be obtained from a real-time clock.
C
C Input variables: none
C Output variables: none
C Procedures called: none
C Common blocks: TIME
C
      INTEGER DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
C Local Variables: none
C
C---------------- increment time ---------------------------------------
      SECOND = SECOND + 1
      IF(SECOND.GE.60)THEN
        SECOND = 0
        MINUTE = MINUTE + 1
        IF(MINUTE.GE.60)THEN
          MINUTE = 0
          HOUR = HOUR + 1
          IF(HOUR.GE.24)THEN
            HOUR = 0
            DAY = DAY + 1
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
C=======================================================================
      SUBROUTINE TIMINC(STEPDN,TIMEUP)
C=======================================================================
C This routine determines if the
C desired duration of the execution has passed (TIMEUP=true), and if the
C simulation timestep has passed since the last execution of the
C simulation subroutine (STEPDN=true).
C
C Input variables: none
C Output variables:
C STEPDN - Logical variable, true if time for another simulation execution.
C TIMEUP - Logical variable, true if test ending time has been reached
C
      LOGICAL STEPDN, TIMEUP
```

```
C
C Procedures called: none
C Common blocks: TIME,PAST
C
      INTEGER DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      INTEGER OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
      COMMON / TIME / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / PAST / OLDDAY,OLDHR,OLDMIN,OLDSEC,STEP
C
C Local Variables:
C
C JTIME  - Integer used to contain time differential in seconds.
C
      INTEGER JTIME
C
C------------------ check for end of program --------------------------
      CALL SBTIME(ENDDAY,ENDHR,ENDMIN,ENDSEC,DAY,HOUR,MINUTE,SECOND,
     & JTIME)
      IF(JTIME.LT.0) THEN
        TIMEUP = .TRUE.
      ELSE
        TIMEUP = .FALSE.
      ENDIF
C------------------ check for end of time step ------------------------
      CALL SBTIME(DAY,HOUR,MINUTE,SECOND,OLDDAY,OLDHR,OLDMIN,OLDSEC,
     & JTIME)
      IF(JTIME.GE.STEP) THEN
        STEPDN = .TRUE.
      ELSE
        STEPDN = .FALSE.
      ENDIF
      RETURN
      END
C=====================================================================
      SUBROUTINE SBTIME(DAY,HOUR,MIN,SEC,MDAY,MHOUR,MMIN,MSEC,DSEC)
C=====================================================================
C This module subtracts two sets of time integers (xxx - Mxxx) and
C puts the result in seconds into DSEC.
C
C Input variables:
C DAY    - Day number of first time from which second time is subtracted.
C HOUR   - Hour of first time.
C MIN    - Minute of first time.
C SEC    - Second of first time.
C MDAY   - Day number of second time which is to be subtraced from first.
C MHOUR  - Hour of second time.
C MMIN   - Minute of second time.
C MSEC   - Second of second time.

C Output variables:
C DSEC - differential in seconds between first and second times.
C
      INTEGER DAY,HOUR,MIN,SEC,MDAY,MHOUR,MMIN,MSEC,DSEC
C
C Procedures called: none
C Common blocks: none
C
C Local Variables:
```

```
C
C CDAY   - Working copy of day from first time.
C CHOUR  - Working copy of hour from first time.
C CMIN   - Working copy of minute from first time.
C CSEC   - Working copy of second from first time.
C DDAY   - time differential, day component.
C DHOUR  - time differential, hour component.
C DMIN   - time differential, minute component.
C
      INTEGER CDAY,CHOUR,CMIN,CSEC,DDAY,DHOUR,DMIN
C
      CDAY = DAY
      CHOUR = HOUR
      CMIN = MIN
      CSEC = SEC
C
      DSEC = CSEC - MSEC
      IF(DSEC.LT.0)THEN
        DSEC = DSEC + 60
        CMIN = CMIN - 1
      ENDIF
C
      DMIN = CMIN - MMIN
      IF(DMIN.LT.0)THEN
        DMIN = DMIN + 60
        CHOUR = CHOUR - 1
      ENDIF
C
      DHOUR = CHOUR - MHOUR
      IF(DHOUR.LT.0)THEN
        DHOUR = DHOUR + 24
        CDAY = CDAY -1
      ENDIF
      DDAY = CDAY - MDAY
C
      DSEC = DSEC + (60*DMIN) + (3600*DHOUR) + (86400*DDAY)
C
      RETURN
      END
C=======================================================================
      SUBROUTINE UPDATE
C=======================================================================
C This routine is intended to emulate a real time I/O task which is
C connected to hardware I/O channels. Instead, it receives input from
C a 'boundary value' file to define changes in sensor readings.
C The following sensors and actuators are supported:
C  EMCS DIGITAL OUTPUTS                      VARIABLE NAME      0        1
C   Supply Fan On/Off                             ON           off       on
C   O.A. Damper Off/Auto                         ECON          off      auto
C   Ventilation Damper Open/Close                VENT         open     close
C   Zone Thermostat Setback On/Off              SETBAK         off       on
C
C  EMCS ANALOG OUTPUTS                                         VARIABLE NAME
C   Control Point Adjustment, Supply Air Temperature (F)          SUPPLY
C   Control Point Adjustment, Zone Temperature (F)                ZONE
C
C  EMCS ANALOG INPUTS                          VARIABLE NAME
C   Temperature, Outside Air (F)                   TOUTA
```

```
C    Temperature, Mixed Air  (F)                     TMIXA
C    Temperature, Supply Air  (F)                    TSUPA
C    Temperature, Return Air  (F)                    TRETA
C    Temperature, Space  (F)                         TZONEA
C    Relative Humidity, Outside Air (%)              RHOUTA
C    Relative Humidity, Return Air (%)               RHRETA
C    Relative Humidity, Space  (%)                   RHZONA
C    Position, Reheat Valve (Btu/s)                  ZHCDEM
C
C  EMCS DIGITAL INPUTS
C    Differential Pressure Switch, Supply Fan   DPSSF
C
C Input variables: none
C Output variables: none
C Procedures called: RELHUM
C Common blocks: UNITS,TIME,ASENSR,COMMND,DSENSR,CPA,STATE1
C
      INTEGER               CLU,FLU,ILU,PLU
      COMMON / UNITS  / CLU,FLU,ILU,PLU
C
      INTEGER              DAY,HOUR,MINUTE.SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME   / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
      REAL                  TOUTA,RHOUTA,TZONEA,   RHZONA,   TRETA,
     &                      RHRETA,   ZHCDEM,    TSUPA    ,TMIXA
      COMMON / ASENSR / TOUTA,RHOUTA,TZONEA(5),RHZONA(5),TRETA(5),
     &                      RHRETA(5),ZHCDEM(5),TSUPA(5),TMIXA(5)
C
      LOGICAL               ON,    ECON,    VENT,    SETBAK
      COMMON / COMMND / ON(5),ECON(5),VENT(5),SETBAK(5)
C
      LOGICAL               DPSSF
      COMMON / DSENSR / DPSSF(5)
C
      REAL                  SUPPLY    ,ZONE
      COMMON /   CPA    / SUPPLY(5),ZONE(5)
C
      REAL                  A
      LOGICAL                       L
      COMMON / STATE1 / A(100),L(100)
C
C Global Variables:
C EVENT  - Logical variable which is true when a command from the EMCS
C            file has been received.
      LOGICAL EVENT
      GLOBAL EVENT
C
C Local Variables:
C ANSWER - character variable to receive answer on debug print question.
C DATE   - day number for the next command event
C DBINIT - logical variable, true if debug print question has been asked.
C DBP    - logical variable, true if debug print is enabled.
C EX     - exponential of ST.
C LEFT   - seconds left before next command event will take place.
C NBLD   - building number used as index in EMCS variable arrays.
C ST     - negative reciprocal of TCSENS.
C TCSENS - time constant of EMCS analog sensors in seconds.
C TIME   - time in hours, minutes, and seconds for the next command event.
```

```
C WAIT   - logical variable, true if command event has been read from
C          command event file and is being stored, False if no command
C          event is stored.
C
      REAL EX,ST,TCSENS
      LOGICAL WAIT,DBP,DBINIT
      INTEGER DATE,TIME(3),LEFT
      CHARACTER*1 ANSWER
C
      PARAMETER (NBLD=1, TCSENS=15.0, ST=(-1.0)/TCSENS)
C
      DATA DBP /.FALSE./, DBINIT/.FALSE./, WAIT/.FALSE./
C
      NAMELIST /DEBUG/ DAY,HOUR,MINUTE,SECOND,TOUTA,RHOUTA,TZONEA,
     &RHZONA,TRETA,RHRETA,ZHCDEM,TSUPA,TMIXA,DPSSF
C
C-----------------------------------------------------------------------
C Read the boundary value file to see if EMCS has sent a command. THe
C command will have a time associated with it which is the time the
C command was given. The command data is not changed until the command
C time. The variable WAIT, if true, indicates that the routine is waiting
C for the command time to arrive before reading the boundary file again.
C
      IF(.NOT.WAIT)THEN
  500   READ(FLU,*,ERR=500,END=900) DATE,TIME,ON(NBLD),ECON(NBLD),
     &                              VENT(NBLD),SETBAK(NBLD),
     &                              SUPPLY(NBLD),ZONE(NBLD)
        WAIT = .TRUE.
      ENDIF
C-----------------------------------------------------------------------
C Determine if the time for the command obtained from the boundary file
C has come. If so, then change the command data in common, and set the
C event flag to indicate a command has occurred. LEFT is the time left
C before the command becomes active.
C
 900  IF(WAIT)THEN
        CALL SBTIME(DATE,TIME(1),TIME(2),TIME(3),DAY,HOUR,MINUTE,SECOND,
     &  LEFT)
        IF(LEFT.LE.0)THEN
          L(1)=ON(NBLD)
          L(2)=ECON(NBLD)
          L(3)=VENT(NBLD)
          L(4)=SETBAK(NBLD)
          A(80)=SUPPLY(NBLD)
          A(81)=ZONE(NBLD)
          WAIT=.FALSE.
          EVENT=.TRUE.
        ELSE
          EVENT=.FALSE.
        ENDIF
      ELSE
        EVENT = .FALSE.
      ENDIF
C-----------------------------------------------------------------------
C Update the sensor data in common with the data from the state variables
C calculated by the emulator model. To simulate sensors with finite time
C constants, an exponential approach is used.
C
```

```
      DPSSF(NBLD) = L(10)
C
      EX=EXP(ST)
      TOUTA =         (1.-EX)*A(01) + EX*TOUTA
      TRETA(NBLD) =   (1.-EX)*A(20) + EX*TRETA(NBLD)
      TZONEA(NBLD)=   (1.-EX)*A(06) + EX*TZONEA(NBLD)
      TSUPA(NBLD) =   (1.-EX)*A(32) + EX*TSUPA(NBLD)
      TMIXA(NBLD) =   (1.-EX)*A(23) + EX*TMIXA(NBLD)
      ZHCDEM(NBLD) = A(62)

      RHOUTA =        (1.-EX)*RELHUM(A(01),A(02),29.921) +EX*RHOUTA
      RHRETA(NBLD) = (1.-EX)*RELHUM(A(20),A(22),29.921) +EX*RHRETA(NBLD)
      RHZONA(NBLD) = (1.-EX)*RELHUM(A(06),A(07),29.921) +EX*RHZONA(NBLD)
C
      IF(.NOT.DBINIT)THEN
        WRITE(1,FMT='(1X,"DEBUG PRINTOUT?")')
        READ(1,FMT='(A1)')ANSWER
        IF(ANSWER.EQ.'Y') DBP = .TRUE.
        DBINIT = .TRUE.
      ENDIF
      IF(DBP) WRITE(1,DEBUG)
      RETURN
      END
C Version 1.28 - FEBRUARY 13, 1985 - W.B. MAY ,NATIONAL BUREAU OF STANDARDS
C=======================================================================
      SUBROUTINE EMODEL
C=======================================================================
C This is the main subroutine for the simulation routines for the EMCS
C algorithm tester. The simulation sections consist of the weather data
C section, the Air Handling Unit simulation, the zone simulation, the
C local heating and cooling equipment simulation, the comfort
C simulation, and the data compilation routines. NOTE: all common block
C variables are described in subroutine PARAMA.
C
C Input variables: none
C Output variables: none
C Procedures called: AHU,CMPILE,COMFRT,CONPAR,ECTROL,LOCAL,PARAMA,WEATHR,
C                    ZONE
C Common blocks: CLIMAT,STATE1
C
      REAL                WOAVG,CWIND,TDBAVG,TDBAMP,SINORG,CSOLR
      COMMON / CLIMAT / WOAVG,CWIND,TDBAVG,TDBAMP,SINORG,CSOLR
C
      REAL                A
      LOGICAL                       L
      COMMON / STATE1 / A(100),L(100)
C
C Local variables
C
C GO     - logical variable; if true, indicates that EMODEL should be
C          executed.
C NOINIT - logical variable; if true, indicates the the simulation
C          parameters have not been initialized.
C TIMEST - The current value to be used for the major simulation time step.
C
      LOGICAL NOINIT,GO
      INTEGER TIMEST
      DATA NOINIT/.TRUE./
```

```
C
C-------------------------------------------------------------------------
C Initialize all non-time-dependent parameters
      IF(NOINIT)THEN
        CALL CONPAR
        CALL PARAMA
        NOINIT = .FALSE.
        CALL WEATHR
        CALL ECTROL(TIMEST,GO)
      ELSE
C-------------------------------------------------------------------------
C Determine if simulation routine is to be executed
C
        CALL ECTROL(TIMEST,GO)
        IF(.NOT.(GO)) RETURN
C-------------------------------------------------------------------------
C Determine current weather parameters for this time step.
C
        CALL WEATHR
C-------------------------------------------------------------------------
C Simulate the air handling unit equipment given return air temperature,
C outdoor air conditions and the state of control signals. This is an
C algebraic/logical model assumming a very small time constant.
C
        CALL AHU
C-------------------------------------------------------------------------
C Simulate a building zone given the supply air conditions and
C environmental conditions. This requires solution of differential
C equations.
C
        CALL ZONE(TIMEST)
C-------------------------------------------------------------------------
C Determine the comfort level in the building zone
C
        CALL COMFRT(TIMEST)
C-------------------------------------------------------------------------
C Compile the data generated for later evaluation
C
      ENDIF
      CALL CMPILE(TIMEST)
C
      RETURN
      END
C=========================================================================
      SUBROUTINE PARAMA
C=========================================================================
C This routine is used to initialize the emulator building simulation
C parameters. It is called once at the beginning of a one season test.
C
C Input variables: none
C Output variables: none
C Procedures called: none
C Common blocks: TIME,CLIMAT,HVAC,HVAC2,PFILES,SHELL,STATE1,UNITS,USE
C
C Description of Common blocks:
C TIME   - contains the current time vector for the simulation.
C CLIMAT - contains parameters which describe climate around the building.
C HVAC   - contains parameters which describe the HVAC system of the building.
```

```
C HVAC2  - contains character data describing the HVAC system of the building.
C PFILES - contains the character names of the files containing parameters.
C SHELL    contains parameters which describe the building shell.
C STATE1 - contains the state variables for the emulator model.
C UNITS  - contains the logical unit assignments for program read and write.
C USE      contains parameters which describe building use schedules.
C
C Common Block Variables:
C
C DAY    - Current day number, first day of test is day 1.
C HOUR   - Current hour of day, from 0 - 23.
C MINUTE - Current minute within the hour, 0 - 59.
C SECOND - Current second within the minute, 0 - 59.
C ENDDAY - Day number on which the test is to stop.
C ENDHR  - Hour of the day on ENDDAY within which test is to stop.
C ENDMIN - Minute within ENDHR within which test is to stop.
C ENDSEC - Second within ENDMIN within which test is to stop.
C
      INTEGER             DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME   / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
C WOAVG  - Average outdoor humidity ratio in lb water/lb dry air.
C CWIND  - Average windspeed, MPH
C TDBAVG - Average outdoor dry bulb temperature, F
C TDBAMP - Diurnal amplitude of outdoor dry bulb temperature, F
C SINORG - Origin of sine wave describing dry bulb temperature variation in
C          seconds after midnight.
C CSOLR  - Peak solar heat gain on horizontal.
C
      REAL                WOAVG,CWIND,TDBAVG,TDBAMP,SINORG,CSOLR
      COMMON / CLIMAT / WOAVG,CWIND,TDBAVG,TDBAMP,SINORG,CSOLR
C
C DUDECK - logical variable which is true for a dual deck system.
C HUMCON - logical variable which is true for a humidity control system.
C RAFAN  - logical variable which is true for a Return Air Fan in system.
C VAV    - logical variable which is true for a Variable Air Volume System.
C VOLSAF - Air handling unit fan rated supply air volume in ft*3/sec.
C CAPLOC - Local equipment capacity for heating in Btu/sec.
C PGAINL - Local equipment controller proportional gain in Btu/sec. F
C TCTHSE - Local equipment controller sensor time constant in sec.
C QRAFAN - Return Air Fan air heating rate in Btu/sec.
C FVAMIN - Minimum ventilation air, fraction of supply air mass
C QSAFAN - Supply Air Fan air heating rate in Btu/sec.
C HCAP1  - Cold Deck Heating Coil Capacity in Btu/sec.
C CCAP1  - Cold Deck Cooling Coil Capacity in Btu/sec.
C DTSETB - Local equipment controller setpoint setback in F
C POWSAF - power requirement of supply fan at rated volume in kW
C POWRAF - power requirement of return fan at rated volume in kW
C GAINIL - Local equipment controller integral gain in Btu/s2 F
C LOCEQT - character description of Local equipment type
C
      LOGICAL            DUDECK,HUMCON,RAFAN,VAV
      COMMON / HVAC /    DUDECK,HUMCON,RAFAN,VAV,VOLSAF,CAPLOC,PGAINL,
     &                   TCTHSE,QRAFAN,FVAMIN,QSAFAN,HCAP1,CCAP1,DTSETB,
     &                   POWSAF,POWRAF,GAINIL
      CHARACTER*10       LOCEQT
      COMMON / HVAC2/    LOCEQT
C
```

```
C HVFILE - Character filename of HVAC equipment parameter file.
C USFILE - Character filename of Building usage parameter file.
C CLFILE - Character filename of climate description parameter file.
C SHFILE - Character filename of building shell parameter file.
C INFILE - Character filename of state vector initialization file.
C SEASON - Character name of weather season to use in test.
C
      CHARACTER*15       HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
      COMMON / PFILES / HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
C
C AREAW  - Area of the building wall exposed to the outside in ft^2.
C HIWALL - Inner wall film coefficient in Btu/s F ft^2.
C HOWALL - Outer wall film coefficient in Btu/s F ft^2.
C LENOW  - thickness of wall outside the mass center in ft.
C LENIW  - thickness of wall inside the mass center in ft.
C KOW    - wall conductivity outside the mass center in Btu/s F ft.
C KIW    - wall conductivity inside the mass center in Btu/s F ft.
C CFUR   - thermal capacity of zone furnishings in Btu/F.
C VOLAIR - volume of air in zone in ft^3.
C CPOW   - specific heat of wall outside the mass center in Btu/lb F.
C RHOOW  - density of wall outside the mass center in lbm/ft^3.
C CPIW   - specific heat of wall inside the mass center in Btu/lb F.
C RHOIW  - density of wall inside the mass center in lbm/ft^3.
C AIREXC - base air infiltration rate without wind and thermal in ACH
C RMGFUR - furnishings moisture absorption mass relative to air
C
      REAL                AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                    VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
      COMMON / SHELL /  AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                    VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C contents of state vector
C A(1): TEMPERATURE of outside air (F)
C A(2): HUMIDITY RATIO of outside air (lbw/lbda)
C A(3): ENTHALPY of outside air (Btu/lb)
C A(4): Windspeed (MPH)
C
C A(6): TEMPERATURE of zone air (F)
C A(7): HUMIDITY RATIO of zone air (lbw/lbda)
C A(8): TEMPERATURE of zone wall interior (F)
C A(9): TEMPERATURE of Zone wall surface  (F)
C A(10): TEMPERATURE of zone glass  (F)
C
C A(20): TEMPERATURE of return air (F)
C A(21): ENTHALPY of return air (BTU/lb)
C A(22): HUMIDITY RATIO of return air (lbw/lbda)
C A(23): TEMPERATURE of mixed air (F)
C A(24): ENTHALPY of mixed air (BTU/lb)
C A(25): HUMIDITY RATIO of mixed air (lbw/lbda)
C A(26): TEMPERATURE of cooling coil discharge air (F)          cold deck
C A(27): ENTHALPY of cooling coil discharge air (BTU/lb)        cold deck
C A(28): HUMIDITY RATIO of cooling coil discharge air (lbw/lbda) cold deck
C A(29): TEMPERATURE of heating coil discharge air (F)          cold deck
C A(30): ENTHALPY of heating coil discharge air (BTU/lb)        cold deck
C A(31): HUMIDITY RATIO of heating coil discharge air (lbw/lbda) cold deck
C A(32): TEMPERATURE of supply air (F)                          cold deck
C A(33): ENTHALPY of supply air (BTU/lb)                        cold deck
C A(34): HUMIDITY RATIO of supply air (lbw/lbda)                cold deck
```

```
C A(35): ENTHALPY of return air plus ventilation air(BTU/lb)
C
C A(51): MASS FLOW RATE of return air (lb/s)
C A(52): MASS FLOW RATE of supply air (lb/s)                    cold deck
C A(53): MASS FLOW RATE of ventilation air (lb/s)
C
C A(60): HEATING POWER  to heating coil (Btu)                   cold deck
C A(61): COOLING POWER  from cooling coil (Btu)                 cold deck
C A(62): HEATING POWER  to local zone equipment (Btu/s)
C A(63): COOLING POWER  from local zone equipment (Btu/s)
C A(64): HEATING ENERGY (requirements) (Btu)
C A(65): COOLING ENERGY (requirements) (Btu)
C A(66): HEATING ENERGY (economizer) (Btu)
C A(67): COOLING ENERGY (economizer) (Btu)
C A(68): HEATING ENERGY (load) (Btu)
C A(69): COOLING ENERGY (load) (Btu)
C A(70): HEATING ENERGY (reheat) (Btu)
C A(71): COOLING ENERGY (recool) (Btu)
C A(72): ELECTRICAL ENERGY to fans (kWh)
C A(73): HEATING ENERGY to local zone equipment (Btu)
C A(74): COOLING ENERGY from local zone equipment (Btu)
C
C A(80): CONTROL POINT ADJUSTMENT supply air temperature (F)
C A(81): CONTROL POINT ADJUSTMENT zone temperature (F)
C A(82): CONTROL POINT ADJUSTMENT outside air damper position (F)
C A(83): CONTROL POINT ADJUSTMENT supply air RH (%)
C
C A(90): COMFORT seconds outside of comfort range
C A(91): COMFORT maximum dry bulb temperature (F)
C A(92): COMFORT minimum dry bulb temperature (F)
C A(93): COMFORT relative humidity at maximum temperature (%)
C A(94): COMFORT relative humidity at minimum temperature (%)
C
C L(01): COMMAND air handling unit on/off
C L(02): COMMAND economizer on/off
C L(03): COMMAND ventilation on/off
C L(04): COMMAND zone setback on/off
C
C L(10): STATUS air handling unit on/off
C L(11): STATUS zone occupied/unoccupied
C
      REAL                 A
      LOGICAL                   L
      COMMON / STATE1 / A(100),L(100)
C
C CLU    - Logical Unit for writes to consola.
C FLU    - Logical Unit for read from EMCS command input file.
C ILU    - Logical Unit for read from console.
C PLU    - Logical Unit for read from parameter files.
C
      INTEGER              CLU,FLU,ILU,PLU
      COMMON / UNITS  / CLU,FLU,ILU,PLU
C
C LSCHED - Index of current building usage state description in arrays.
C NSCHED - Number of possible states in schedule arrays.
C TSCHED - Start time of LSCHED'th building usage state in hours past
C          midnight.
C SENSIB - Equipment sensible heat gain for LSCHED'th building usage state
```

```
C           in Btu/s.
C LATENT - Equipment moisture gain for LSCHED'th building usage state
C           in lbm/s.
C PESENS - Sensible heat gain from people for LSCHED'th building usage state
C           in Btu/s.
C PELATE - Moisture gain from people for LSCHED'th building usage state
C           in lbm/s
C
      PARAMETER (NSCHED = 10)
      INTEGER        LSCHED
      REAL                     TSCHED         ,SENSIB,PESENS, LATENT,PELATE
      COMMON / USE / LSCHED,TSCHED(NSCHED),SENSIB(NSCHED),LATENT(NSCHED)
     &                                      ,PESENS(NSCHED),PELATE(NSCHED)
C
C Local Variables:
C AINIT  - Temporary value to hold state vector initializer read from
C           initialization file.
C BTUSPW - Conversion factor, Btu per Watt.
C BULB   - Indicates type of lighting in describing building schedule.
C CHMAX  - Time of maximum outdoor dry bulb temperature, hrs from midnight.
C CTMAX  - Maximum daily outdoor temperature for a particular season in F.
C CTMIN  - Minimum daily outdoor temperature for a particular season in F.
C EQUIPL - Equipment moisture gain for a schedule period in lb water/hr.
C EQUIPS - Electric power used by any equipment in zone for a schedule period
C           in Watts.
C INDEX  - Temporary value to hold state vector index read from initialization
C           file.
C LIGHT  - Lighting power in the zone for a schedule period in watts.
C LINIT  - Temporary variable to hold logical state vector initializer read
C           from initialization file.
C NACCAT - Number of possible activity categories for occupants of zone.
C NACTIV - Array containing number of occupants in NACCAT activity categories
C           for a schedule period.
C OCCLAT - Occupant moisture gain per person for a schedule period in
C           lb water/hour.
C OCCSEN - Occupant sensible heat gain per person for a schedule period in
C           Btu/hr.
C RE     - ratio of radiated energy to total equipment energy for a schedule
C           period.
C RL     - ratio of radiated energy to total lighting energy for a given
C           lighting type.
C RP     - ratio of radiated energy tc total sensible heat of people.
C SEASIN - temporary variable to hold season name from climate file for
C           comparison with desired season in variable SEASON.
C TOLAT  - total moisture gain from occupants for a schedule period in lb.
C           water/hour.
C TOSEN  - total sensible heat gain of occupants for a schedule period in
C           Btu/hour.
C
      REAL EQUIPL,EQUIPS,RE,RL,RP,TOLAT,TOSEN
      PARAMETER (NACCAT = 5,BTUSPW = 9.47817E-4, RP = 0.4)
      LOGICAL LINIT
      INTEGER NACTIV(NACCAT),INDEX
      REAL AINIT,CHMAX,CTMAX,CTMIN,OCCSEN(NACCAT),OCCLAT(NACCAT),LIGHT
      CHARACTER*1 BULB
      CHARACTER*5 SEASIN
C
      DATA OCCSEN / 230., 255., 315., 345., 345. /
```

```
      DATA OCCLAT / 0.20, 0.27, 0.34, 0.46, 0.74 /
C
C          Occupant sensible and moisture gains:
C     (from ASHRAE fundamentals, 1981, p 26.25 and Threlkeld, page 363)
C                                               sensible BTU/hr  moisture LB/hr
C  1. seated, very light work                       230.             0.20
C  2. seated, light work, typing                    255.             0.27
C  3. standing, light work                          315.             0.34
C  4. light bench work                              345.             0.46
C  5. walking, 3 mph, light machine work            345.             0.74
C
C-----------------------------------------------------------------------------
C Initialize State Variables
C
      OPEN(PLU,FILE=INFILE,STATUS='OLD',ERR=994)
      DO 250 I = 1,100
       READ(PLU,*,END=280,ERR=260)INDEX,AINIT
       A(INDEX) = AINIT
  250 CONTINUE
  260 DO 270 J = 1,100
       READ(PLU,*,END=280,ERR=993)INDEX,LINIT
       L(INDEX) = LINIT
  270 CONTINUE
  280 CLOSE(PLU)
C-----------------------------------------------------------------------------
C Read Building Use parameters
C
      OPEN(PLU,FILE=USFILE,STATUS='OLD',ERR=998)
      LSCHED = 0
      READ(PLU,*,END=500)
      DO 400 I = 1,NSCHED
C
C TSCHED in hours, LIGHT and EQUIPS in W, EQUIPL in lb water/hr
C
       READ(PLU,*,END=500) TSCHED(I),LIGHT,BULB,EQUIPS,RE,EQUIPL,
     &                     (NACTIV(K),K=1,NACCAT)
C
C Determine lighting type, I = incandescent, F = fluorescent
C
       IF(BULB.EQ.'I') THEN
         RL = 0.8
       ELSE IF(BULB.EQ.'F') THEN
         RL = 0.5
       ELSE
         RL = 0.5
       ENDIF
C
       LSCHED = LSCHED + 1
       TOSEN = 0.0
       TOLAT = 0.0
       DO 300 J = 1,NACCAT
        TOSEN = TOSEN + NACTIV(J) * OCCSEN(J)
        TOLAT = TOLAT + NACTIV(J) * OCCLAT(J)
  300  CONTINUE
       SENSIB(I) = ((1-RL)*LIGHT + (1-RE)*EQUIPS) * BTUSPW
       LATENT(I) = EQUIPL / 3600.
       PESENS(I) = (1-RP)*TOSEN / 3600.
       PELATE(I) = TOLAT / 3600.
```

```
  400 CONTINUE
C
C-----------------------------------------------------------------------
C Read Building Shell Parameters
C
  500 OPEN(PLU,FILE=SHFILE,STATUS='OLD',ERR=999)
      READ(PLU,*) AREAW
      READ(PLU,*) HIWALL
      READ(PLU,*) HOWALL
      READ(PLU,*) LENOW
      READ(PLU,*) LENIW
      READ(PLU,*) KOW
      READ(PLU,*) KIW
      READ(PLU,*) CFUR
      READ(PLU,*) VOLAIR
      READ(PLU,*) CPOW
      READ(PLU,*) RHOOW
      READ(PLU,*) CPIW
      READ(PLU,*) RHOIW
      READ(PLU,*) AIREXC
      READ(PLU,*) RMGFUR
      CLOSE(PLU)
C
C-----------------------------------------------------------------------
C Read HVAC system Parameters
C
      OPEN(PLU,FILE=HVFILE,STATUS='OLD',ERR=997)
      READ(PLU,*) VOLSAF
      READ(PLU,*) POWSAF
      READ(PLU,*) POWRAF
      READ(PLU,*) LOCEQT
      READ(PLU,*) CAPLOC
      READ(PLU,*) PGAINL
      READ(PLU,*) GAINIL
      READ(PLU,*) TCTHSE
      READ(PLU,*) VAV
      READ(PLU,*) RAFAN
      READ(PLU,*) QRAFAN
      READ(PLU,*) FVAMIN
      READ(PLU,*) DUDECK
      READ(PLU,*) QSAFAN
      READ(PLU,*) HUMCON
      READ(PLU,*) HCAP1
      READ(PLU,*) CCAP1
      READ(PLU,*) DTSETB
      CLOSE(PLU)
C-----------------------------------------------------------------------
C Read Climate Parameters
C
      OPEN(PLU,FILE=CLFILE,STATUS='OLD',ERR=996)
      READ(PLU,*) SEASIN
  600 READ(PLU,*,END=995) SEASIN
      READ(PLU,*) CTMAX
      READ(PLU,*) CTMIN
      READ(PLU,*) CHMAX
      READ(PLU,*) WOAVG
      READ(PLU,*) CSOLR
      READ(PLU,*) CWIND
```

```
      IF(SEASIN(1:5).NE.SEASON(1:5)) GO TO 600
      CLOSE(PLU)
      TDBAVG = (CTMAX + CTMIN) /2.
      TDBAMP = CTMAX - TDBAVG
      SINORG = (CHMAX-30.)*3600.
C
      RETURN
C---------ERRORS------------------------------------------------------------
  993 WRITE(CLU,7)I+J
    7 FORMAT(1X,'unable to read INITIALIZATION file at line',I6)
      STOP
  994 WRITE(CLU,6)
    6 FORMAT(1X,'unable to open INITIALIZATION file')
      STOP
  995 WRITE(CLU,5)
    5 FORMAT(1X,'unable to locate requested season in CLIMATE file')
      STOP
  996 WRITE(CLU,4)
    4 FORMAT(1X,'unable to open CLIMATE parameter file')
      STOP
  997 WRITE(CLU,3)
    3 FORMAT(1X,'unable to open HVAC parameter file')
      STOP
  998 WRITE(CLU,2)
    2 FORMAT(1X,'unable to open USE parameter file')
      STOP
  999 WRITE(CLU,1)
    1 FORMAT(1X,'unable to open SHELL parameter file')
      STOP
      END
C=========================================================================
      SUBROUTINE WEATHR
C=========================================================================
C This routine is used to determine the current weather conditions. Wind
C and humidity ratio are considered constants. Outdoor temperature is
C assummed to vary sinusoidally with a period of 24 hours.
C
C AS of 2-15-85 there is no solar energy section.
C
C Input variables: none
C Output variables: none
C Procedures called: SINUSO
C Common blocks: STATE1, CLIMAT, TIME
C
      REAL                A
      LOGICAL                       L
      COMMON / STATE1 / A(100),L(100)
C
      INTEGER            DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME   / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
      COMMON / CLIMAT / WOAVG,CWIND,TDBAVG,TDBAMP,SINORG,CSOLR
C
C Local Variables:
C
C PERIOD - Period of sinusoidal temperature variation in seconds.
C RTIME  - Current time in seconds past midnight.
C TOUTA  - Outside Air dry bulb temperature in F
```

```
C VWIND  - Wind velocity in MPH.
C WOUTA  - Outside air humidity ratio in lb water / lb dry air
C
      REAL PERIOD,RTIME
      PARAMETER (PERIOD=86400.)
      EQUIVALENCE ( A(1),TOUTA ),( A(2),WOUTA ),(A(04),VWIND)
C
      RTIME = HOUR*3600 + MINUTE*60 + SECOND
      CALL SINUSO(TDBAVG,PERIOD,TDBAMP,RTIME,SINORG,TOUTA)
      WOUTA = WOAVG
      VWIND = CWIND
      RETURN
      END
C=========================================================================
      SUBROUTINE SINUSO(TDBAVG,PERIOD,AMP,RTIME,RORIG,T)
C=========================================================================
C Returns a point on an approximate sinusoidal curve as a function of
C time. The curve shape is based on table data in SHAPE.

C Input variables:
C TDBAVG - the average value or baseline of the curve.
C PERIOD - contains the period of the wave in seconds.
C AMP    - the amplitude of the curve with reference to the baseline.
C RTIME  - the current time in seconds since midnight.
C RORIG  - the origin of the wave in seconds since midnite.

C Output variables:
C T      - the current value of the wave at time RTIME.
C
C Procedures called: none
C Common blocks: none
C
C Local Variables:
C
C ICURP2 - Index into SHAPE table for smallest time larger than current time.
C ICURPE - Index into SHAPE table for largest time smaller than current time.
C NPER   - Number of whole curve periods passed by current time.
C NPIECE - Number of entries in curve look-up table.
C RANGE  - Difference between two table entries for times on either side
C          of current time.
C RCURPE - Real number of curve pieces past origin for current time.
C RNPPS  - Real number of curve pieces per second.
C SHAPE  - Curve look-up table containing normalized curve values for one
C          period of oscillation equally divided into NPIECE parts.
C
      PARAMETER (NPIECE = 24)
      REAL SHAPE(NPIECE)
      DATA SHAPE/+0.0000,+0.2588,+0.5000,+0.7071,+0.8660,+0.9659,
     &           +1.0000,+0.9659,+0.8660,+0.7071,+0.5000,+0.2588,
     &           -0.0000,-0.2588,-0.5000,-0.7071,-0.8660,-0.9659,
     &           -1.0000,-0.9659,-0.8660,-0.7071,-0.5000,-0.2588/
C
      RNPPS = FLOAT(NPIECE) / PERIOD
      RCURPE = RNPPS * (RTIME - RORIG)
      ICURPE = AINT (RCURPE)
      NPER = ICURPE / NPIECE
      ICURPE = ICURPE - NPER * (NPIECE)
      RCURPE = RCURPE - NPER * (NPIECE)
```

```
      ICURP2 = ICURPE + 2
      IF(ICURP2.GT.NPIECE) ICURP2 = ICURP2 - NPIECE
      RANGE = (SHAPE(ICURP2) - SHAPE(ICURPE+1))
      T = ((RCURPE - ICURPE) * RANGE + SHAPE(ICURPE+1)) * AMP + TDBAVG
      RETURN
      END
C=======================================================================
      SUBROUTINE AHU
C=======================================================================
C This routine contains a steady state model of a generalized air handling
C unit.
C Input variables: none
C Output variables: none
C Procedures called: CPAIR,DEWPT,ENTHAL,HUMRAT,RHOAIR,TDBSAT.TEMP
C Common blocks: HVAC,HVAC2,STATE1
C
      LOGICAL          DUDECK,HUMCON,RAFAN,VAV
      COMMON / HVAC /  DUDECK,HUMCON,RAFAN,VAV,VOLSAF,CAPLOC,PGAINL,
     &                 TCTHSE,QRAFAN,FVAMIN,QSAFAN,HCAP1,CCAP1,DTSETB,
     &                 POWSAF,POWRAF,GAINIL
      CHARACTER*10     LOCEQT
      COMMON / HVAC2/  LOCEQT
C
      REAL                A
      LOGICAL                       L
      COMMON / STATE1  / A(100),L(100)
C
C Local Variables:
C
C AHSTAT  - STATUS air handling unit on/off
C DELTA   - Difference between supply air setpoint and fan entering air T.
C DTRF    - Temperature differential across the return fan.
C DTSF    - Temperature differential across the supply fan.
C ECONO   - COMMAND economizer on/off
C HCA1    - ENTHALPY of cooling coil discharge air (BTU/lb)        cold deck
C HHA1    - ENTHALPY of heating coil discharge air (BTU/lb)        cold deck
C HMA     - ENTHALPY of mixed air (BTU/lb)
C HMC     - ENTHALPY of return air plus ventilation air(BTU/lb)
C HMO     - Enthalpy of mixed air if outside air dampers are open (Btu/lb)
C HOUTA   - ENTHALPY of outside air (Btu/lb)
C HRA     - ENTHALPY of return air (BTU/lb)
C HSA1    - ENTHALPY of supply air (BTU/lb)                        cold deck
C MRA     - MASS FLOW RATE of return air (lb/s)
C MSA1    - MASS FLOW RATE of supply air (lb/s)                    cold deck
C PRESS   - Atmospheric pressure (assummed constant) in inches of Hg.
C QCOOL1  - COOLING POWER  from cooling coil (Btu)                 cold deck
C QHEAT1  - HEATING POWER  to heating coil (Btu)                   cold deck
C TCA1    - TEMPERATURE of cooling coil discharge air (F)          cold deck
C TDPMA   - Dewpoint temperature of mixed air (F)
C THA1    - TEMPERATURE of heating coil discharge air (F)          cold deck
C TMA     - TEMPERATURE of mixed air (F)
C TMC     - Temperature of return air plus ventilation air (F)
C TMO     - Temperature of mixed air if outside air dampers are open (F)
C TOUTA   - TEMPERATURE of outside air (F)
C TRA     - TEMPERATURE of return air (F)
C TSA1    - TEMPERATURE of supply air (F)                          cold deck
C TSET    - CONTROL POINT ADJUSTMENT supply air temperature (F)
C TZA     - TEMPERATURE of zone air (F)
```

```
C UNITON - COMMAND air handling unit on/off
C VENOFF - COMMAND ventilation on/off
C WCA1   - HUMIDITY RATIO of cooling coil discharge air (lbw/lbda) cold deck
C WHA1   - HUMIDITY RATIO of heating coil discharge air (lbw/lbda) cold deck
C WMA    - HUMIDITY RATIO of mixed air (lbw/lbda)
C WMC    - Humidity ratio of return air plus ventilation air (lbw/lbda)
C WMO    - Humidity ratio of mixed air if outside air dampers are open (lbw/lbda
C WOUTA  - HUMIDITY RATIO of outside air (lbw/lbda)
C WRA    - HUMIDITY RATIO of return air (lbw/lbda)
C WSA1   - HUMIDITY RATIO of supply air (lbw/lbda)                   cold deck
C WZA    - HUMIDITY RATIO of zone air (lbw/lbda)
C
      REAL MRA,MSA1
      EQUIVALENCE ( A(01),TOUTA  ),( A(02),WOUTA  ),( A(20),TRA     )
      EQUIVALENCE ( A(22),WRA    ),( A(80),TSET   ),( A(24),HMA     )
      EQUIVALENCE ( A(06),TZA    ),( A(07),WZA    ),( A(33),HSA1    )
      EQUIVALENCE ( A(32),TSA1   ),( A(34),WSA1   ),( A(52),MSA1    )
      EQUIVALENCE ( A(21),HRA    ),( A(51),MRA    ),( A(25),WMA     )
      EQUIVALENCE ( A(26),TCA1   ),( A(28),WCA1   ),( A(27),HCA1    )
      EQUIVALENCE ( A(29),THA1   ),( A(31),WHA1   ),( A(30),HHA1    )
      EQUIVALENCE ( A(03),HOUTA  ),( A(23),TMA    ),( A(35),HMC     )
      EQUIVALENCE ( A(60),QHEAT1 ),( A(61),QCOOL1 )
C
      LOGICAL UNITON,VENOFF,ECONO,AHSTAT
      EQUIVALENCE ( L(01),UNITON ),( L(03),VENOFF ),( L(02),ECONO   )
      EQUIVALENCE ( L(10),AHSTAT )
C
      PARAMETER (PRESS = 29.921)
C
C-----------------------------------------------------------------------
C Air handling unit is operating - determine air flow rates
C
      IF(UNITON) THEN
        AHSTAT = .TRUE.
        HOUTA = ENTHAL(TOUTA,WOUTA,PRESS)
        IF(VAV) THEN
C***********************************************************************
          STOP 'NO VAV IMPLEMENTED'
C***********************************************************************
        ELSE
          MSA1 = VOLSAF * RHOAIR(WSA1,TSET)
C***********************************************************************
          MRA = MSA1
C***********************************************************************
        ENDIF
C
C-----------------------------------------------------------------------
C Determine heating of air by return fan
C
        IF(RAFAN) THEN
          DTRF = QRAFAN / (MRA * CPAIR(WZA) )
          TRA = TZA + DTRF
        ELSE
          TRA = TZA
        ENDIF
        WRA = WZA
        HRA = ENTHAL(TRA,WRA,PRESS)
C
```

```
      C-----------------------------------------------------------------
      C Determine mixed air conditions
      C
      C---------state of air for outside air dampers closed---------------
      C---------minimum outside air dampers closed------------------------
              IF(VENOFF) THEN
                HMC = HRA
                WMC = WRA
                TMC = TRA
      C----------minimum outside air dampers open-------------------------
              ELSE
                HMC = (FVAMIN * HOUTA) + (1-FVAMIN)*HRA
                WMC = (FVAMIN * WOUTA) + (1-FVAMIN)*WRA
                TMC = TEMP(HMC,WMC,PRESS)
              ENDIF
      C----------determine mixed air temperature--------------------------
              IF(.NOT.DUDECK)THEN
      C---------single deck-----------------------------------------------
      C---------supply fan temperature rise-------------------------------
                DTSF = QSAFAN / (MSA1 * CPAIR(WRA))
                DELTA = TSET - DTSF
                IF(.NOT.HUMCON)THEN
      C-----------economizer enabled-determine damper position------------
                  IF(ECONO) THEN
                    HMO = HOUTA
                    TMO = TOUTA
                    WMO = WOUTA
      C---------------outside air damper should be closed----------------
                    IF(TMC.LE.DELTA) THEN
                      TMA = TMC
                      WMA = WMC
                      HMA = HMC
      C---------------outside air damper should be open------------------
                    ELSE IF(TMC.GT.DELTA.AND.TMO.GT.DELTA) THEN
                      TMA = TMO
                      WMA = WMO
                      HMA = HMO
      C---------------outside air damper should be part open-------------
                    ELSE IF(TMC.GT.DELTA.AND.TMO.LE.DELTA) THEN
                      WMA = (WMC - WMO)/(TMC - TMO)*(DELTA-TMO) + WMO
                      TMA = DELTA
                      HMA = ENTHAL(TMA,WMA,PRESS)
                    ENDIF
                  ELSE
                    HMA = HMC
                    WMA = WMC
                    TMA = TMC
                  ENDIF
      C*****************************************************C
      C  damper position control or report could be installed C
      C*****************************************************C
      C-----------------------------------------------------------------
      C Determine heating or cooling required
      C
      C--------------no heating or cooling required------------------------
                  IF(TMA.EQ.DELTA) THEN
                    TCA1 = TMA
                    WCA1 = WMA
```

```
              HCA1 = HMA
              THA1 = TMA
              WHA1 = WMA
              HHA1 = HMA
              QHEAT1 = 0.0
              QCOOL1 = 0.0
C-------------heating required ---------------------------------------
            ELSE IF(TMA.LT.DELTA) THEN
              THA1 = DELTA
              WHA1 = WMA
              HHA1 = ENTHAL(THA1,WHA1,PRESS)
              QHEAT1 = MSA1 * (HHA1 - HMA)
              IF(QHEAT1.GT.HCAP1) THEN
                HHA1 = HCAP1/MSA1 +HMA
                THA1 = TEMP(HHA1,WHA1,PRESS)
                QHEAT1 = HCAP1
              ENDIF
              HCA1 = HHA1
              TCA1 = THA1
              WCA1 = WHA1
              QCOOL1 = 0.0
C-------------- cooling required---------------------------------------
            ELSE IF(TMA.GT.DELTA) THEN
              THA1 = TMA
              WHA1 = WMA
              HHA1 = HMA
              QHEAT1 = 0.0
              TCA1 = DELTA
              TDPMA = DEWPT(WMA,PRESS)
              IF(TCA1.GT.TDPMA) THEN
                HCA1 = ENTHAL(TCA1,WMA,PRESS)
                WCA1 = WMA
              ELSE
                WCA1 = HUMRAT(TCA1,100.0,PRESS)
                HCA1 = ENTHAL(TCA1,WCA1,PRESS)
              ENDIF
              QCOOL1 = MSA1 * (HMA - HCA1)
              IF(QCOOL1.GT.CCAP1) THEN
                HCA1 = HMA - CCAP1/MSA1
                QCOOL1 = CCAP1
                IF(HCA1.GT.ENTHAL(TDPMA,WMA,PRESS)) THEN
                  TCA1 = TEMP(HCA1,WMA,PRESS)
                  WCA1 = WMA
                ELSE
                  TCA1 = TDBSAT(HCA1,PRESS)
                  WCA1 = HUMRAT(TCA1,100.0,PRESS)
                ENDIF
              ENDIF
            ENDIF
C-------------determine supply air temperatures----------------------
            WSA1 = WCA1
            TSA1 = TCA1 + DTSF
            HSA1 = ENTHAL(TSA1,WSA1,PRESS)
          ELSE
            STOP 'HUMIDITY CONTROL NOT IMPLEMENTED'
          ENDIF
        ELSE
          STOP 'DUAL DECK NOT IMPLEMENTED'
```

```
            ENDIF
C
C-----------------------------------------------------------------------
C Air Handling Unit Off
C
        ELSE
          AHSTAT = .FALSE.
          MSA1 = 0.
          MRA = MSA1
          TSA1 = TZA
          TRA = TZA
          TMA = TZA
          WSA1 = WZA
          WRA = WZA
          WMA = WZA
        ENDIF
        RETURN
        END
C=======================================================================
        SUBROUTINE LOCAL(TIMEST)
C=======================================================================
C This routine contains a simple model of the local heating and/or
C cooling equipment in the zone. It is assummed that the change in output
C of the local equipment is proportional to the change in error between
C the local controller setpoint and the room air temperature and also is
C proportional to the integral of the error (error*timestep). Local
C equipment may ba of several types as configured with the variables in
C common blocks HVAC and HVAC2.
C
C Input variables:
C TIMEST - Tha current value to be usad for the simulation time step.
C
C Output variables: none
C Procedures called: ENTHAL
C Common blocks: HVAC,HVAC2,STATE1
C
        LOGICAL           DUDECK,HUMCON,RAFAN,VAV
        COMMON / HVAC /   DUDECK,HUMCON,RAFAN,VAV,VOLSAF,CAPLOC,PGAINL,
       &                  TCTHSE,QRAFAN,FVAMIN,QSAFAN,HCAP1,CCAP1,DTSETB,
       &                  POWSAF,POWRAF,GAINIL
        CHARACTER*10      LOCEQT
        COMMON / HVAC2/   LOCEQT
C
        REAL                A
        LOGICAL                    L
        COMMON / STATE1 / A(100),L(100)
C
C Local Variables:
C
C DQAIR   - Required change in heat contained in supply air entering zone to
C           reduce zone temperature error. (Btu/s)
C DQENT   - Required change in total supplied heat entering zone to reduce
C           zone temperature error. Equals DQAIR + DQLOC.(Btu/s)
C DQLOC   - Required change in auxilliary heat added to supply air by local
C           zone equipment to reduce zone temperature error.(Btu/s)
C ELOCC   - COOLING ENERGY from local zone equipment (Btu)
C ELOCH   - HEATING ENERGY to local zone equipment (Btu)
C ERROR   - difference between local zone setpoint and zone temperature at
```

```
C              current time.
C ERRORO - difference between local zone setpoint and zone temperature at
C              previous timestep.
C EX     - Transition ratio for local controller temperature sensor
C HSA1   - ENTHALPY of supply air (BTU/lb)                          cold deck
C HSA2   - Enthalpy of hot deck (if existing) supply air (Btu/lb)
C HZA    - Enthalpy of zone air (Btu/lb)
C MRA    - MASS FLOW RATE of return air (lb/s)
C MSA1   - MASS FLOW RATE of supply air (lb/s)                      cold deck
C MSA2   - Mass flow rate of hot deck (if existing) supply air (lb/s)
C PRESS  - Atmospheric pressure (assummed constant) in inches of Hg.
C QAIR   - Heat energy entering zone space through supply air.
C QLOCH  - HEATING POWER  to local zone equipment (Btu/s)
C SETBAK - COMMAND zone setback on/off
C TCTHSE - Time constant of temperature sensor in local controller (s).
C TREF   - Current local zone setpoint temperature (F).
C TSETLO - CONTROL POINT ADJUSTMENT zone temperature (F)
C TTHSE  - Temperature measured by temperature sensor in local controller (s).
C TZA    - TEMPERATURE of zone air (F)
C WZA    - HUMIDITY RATIO of zone air (lbw/lbda)
C
C       REAL MRA,MSA1,MSA2
        LOGICAL SETBAK
        EQUIVALENCE ( A(81),TSETLO ),( L(04),SETBAK ),( A(52),MSA1    )
        EQUIVALENCE ( A(06),TZA    ),( A(07),WZA    ),( A(33),HSA1    )
        EQUIVALENCE ( A(51),MRA    ),( A(62),QLOCH  )
        EQUIVALENCE ( A(73),ELOCH  ),( A(74),ELOCC  )
C
        PARAMETER (PRESS = 29.921)
        DATA  TTHSE/75./,ERRORO/0./
C
        IF(SETBAK)THEN
          TREF = TSETLO - DTSETB
        ELSE
          TREF = TSETLO
        ENDIF
        EX = EXP(-TIMEST/TCTHSE)
        TTHSE = (1-EX)*TZA + EX*TTHSE
        ERROR = TREF - TTHSE
        DQENT = PGAINL * (ERROR-ERRORO) + GAINIL * ERROR * TIMEST
        ERRORO = ERROR
        HZA = ENTHAL(TZA,WZA,PRESS)
C         THE FOLLOWING VALUES ARE FOR DUAL DECK (NOT IMPLEMENTED)
          MSA2 = 0.0
          HSA2 = 0.0
C
        QAIR  = (HSA1 * MSA1) + (HSA2 * MSA2) - (HZA * MRA)
C
C--dual deck---------------------------------------------------------------
        IF(LOCEQT.EQ.'DUAL DECK') THEN
          STOP 'DUAL DECK NOT IMPLEMENTED'
C--VAV---------------------------------------------------------------------
        ELSE IF(VAV) THEN
          DQAIR = DQENT
          QAIR = QAIR + DQAIR
          MSA1 = QAIR / (HSA1 - HZA)
          PAUSE 'ZONE VAV BOX AIR FLOW LIMITS NOT IMPLEMENTED'
C--constant volume reheat-------------------------------------------------
```

```
      ELSE IF(LOCEQT.EQ.'REHEAT') THEN
        DQLOC = DQENT
        QLOCH = QLOCH + DQLOC
        IF(QLOCH.GT.CAPLOC) QLOCH = CAPLOC
        IF(QLOCH.LE.0.) QLOCH = 0.
        ELOCH = ELOCH + QLOCH * TIMEST
      ELSE
        STOP 'LOCAL EQUIPMENT TYPE NOT IMPLEMENTED'
      ENDIF
      RETURN
      END
C=========================================================================
      SUBROUTINE ZONE(TIMEST)
C=========================================================================
C ZONE solves for the values of the variables (in array Y) describing a
C building zone. The current values in Y are used as the starting point.
C New values of Y at one time step in the future are calculated and
C replace the old values of Y.
C
C Input variables:
C TIMEST - The current value to be used for the major simulation time step.
C
      INTEGER TIMEST
C
C Output variables: none
C Procedures called: EULER,SURFAC,ZONE,ZONEFN,LOCAL
C
      EXTERNAL ZONEFN
C
C Common blocks: STATE1,TIME
C
      REAL             A
      LOGICAL                    L
      COMMON / STATE1 / A(100),L(100)
C
      INTEGER          DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME   / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
C Local Variables:
C
C ADSTEP - Logical variable which if true, enables the use of the algorithm
C          to internally determire a minimum Euler timestep. If false, a
C          fixed Euler timestep is used.
C DELMAX - The maximum change in state variable allowed for any of the
C          differential equations describing the zone within one Euler
C          timestep in degrees F. Theoretically, each differential equation
C          should have its own DELMAX. A simplification of this is used.
C EISMIN - Minimum Euler timestep which can be used.
C EISTEP - Current Euler timestep to be used in solving the differential
C          equations. determined from dt = dYmax / (dY/dt), where dYmax is
C          DELMAX.
C EUSTEP - Euler timestep to be used if ADSTEP is false.
C FMAX   - Magnitude of the largest dY/dt for all of the first order
C          differential equations used to describe
C          the zone.
C ITIME  - A time value which is a combination of the current minutes and
C          seconds of the time of day. Used to determine if time-dependent
C          functions should be recalculated.
```

```
C NDEQNS - Number of variables in the zone model state vector which are
C          defined by differential equations. These variables will be the
C          first NDEQNS variables in the vector. The remaining variables
C          will be defined by algebraic equations.
C          in the state vector
C NITER  - Number of iterations required to solve the zone differential
C          equations for one major simulation timestep.
C NSTVAR - Total number of variables in the zone model state vector.
C STEP   - Floating point equivalent of the integer major timestep in the
C          simulation.
C Y      - Zone model state vector which is a subset of the simulation
C          state vector. Elements are defined as:
C          Y(1): zone dry bulb temperature (F)
C          Y(2): zone humidity ratio
C          Y(3): zone internal wall temperature at mass center (F)
C          Y(4): zone interior wall temperature (F)
C          Y(5): zone interior glass temperature (F)
C
      LOGICAL ADSTEP
      PARAMETER (ADSTEP = .TRUE.,EUSTEP = 0.1)
      PARAMETER (NSTVAR = 5,NDEQNS = 3,DELMAX = 0.10, EISMIN = 0.001)
      INTEGER ITIME
      REAL Y(NSTVAR)
      EQUIVALENCE ( A(6),Y(1) )
C
C-------------determine Euler time step------------------------------
C
      IF(ADSTEP)THEN
        FMAX = 0.0
        DO 900 I = 1,NDEQNS
         FMAX = AMAX1( ABS(ZONEFN(I,SECOND,NSTVAR,Y)) , FMAX )
         IF(FMAX.LE.1.E-8) FMAX = 1.E-8
  900   CONTINUE
        STEP = FLOAT(TIMEST)
        EISTEP = DELMAX / FMAX
        EISTEP = AMIN1( EISTEP, STEP )
        EISTEP = AMAX1( EISTEP, EISMIN )
        NITER  = AINT( STEP / EISTEP )
        EISTEP = STEP / FLOAT(NITER)
      ELSE
        NITER = TIMEST / EUSTEP
        EISTEP = EUSTEP
      ENDIF
C----------loop through internal iterations--------------------------
C
      ITIME = SECOND + MINUTE*60
      DO 1000 I = 1,NITER
C
C----------solve NDEQNS simultaneous linear differential equations------
C
        CALL EULER(NDEQNS,ITIME,Y,EISTEP,ZONEFN,NSTVAR)
C
C----------solve algebraic equations for wall and glass surfaces--------
C
        CALL SURFAC(NDEQNS,NSTVAR,Y,ITIME)
C---------------------------------------------------------------------
C Simulate the local zone space heating and cooling equipment. This is
C an algebraic approximation.
```

```
C
       CALL LOCAL(EISTEP)
 1000 CONTINUE
C
      RETURN
      END
C=====================================================================
      SUBROUTINE EULER(NDEQNS,TIME,Y,TIMEST,FNC,NSTVAR)
C=====================================================================
C This subroutine advances the system of NDEQNS ODE'S described by:
C dY(I)/dt = FNC(I,t,Y(1),Y(2),...Y(NDEQNS) by one step of size
C TIMEST using the explicit Euler method. Y contains the dependent
C variables and is overwritten with the new values.
C
C Input variables:
C NDEQNS - Number of differential equations to be advanced.
C TIME   - A time value which must be distinct from the time value on
C          the previous call to EULER.
C Y      - contains the current values of the dependent variables.
C TIMEST - The Euler timestep size.
C FNC    - The address of the function describing dY/dt.
C NSTVAR - dimension of the Y array. Not all elements may be advanced.
C
      REAL Y(NSTVAR),TIMEST
      INTEGER TIME
C
C Output variables: none
C Y      - contains the new value of the dependent variables.
C
C Procedures called: FNC
C
      EXTERNAL FNC
C
C Common blocks: none
C
C----------advance dependent variables--------------------------------
C
        DO 900 J = 1,NDEQNS
          Y(J) = Y(J) + TIMEST * FNC(J,TIME,NSTVAR,Y)
  900   CONTINUE
C
C----------the independent variable, time is advanced externally------
C
      RETURN
      END
C=====================================================================
      SUBROUTINE SURFAC(NDEQNS,NSTVAR,Y,TIME)
C=====================================================================
C This subroutine is used to calculate the next value of the zone state
C variables which can be calculated from an algebraic equation.
C
C Input variables:
C NDEQNS - Number of variables in the zone model state vector which are
C          defined by differential equations. These variables will be the
C          first NDEQNS variables in the vector. The remaining variables
C          will be defined by algebraic equations.
C          in the state vector
C NSTVAR - Total number of variables in the zone model state vector.
```

```
C Y       - contains the current values of the dependent variables.
C TIME    - A time value which is a combination of the current minutes and
C           seconds of the time of day. Used to determine if time-dependent
C           functions should be recalculated.
C
      REAL Y(NSTVAR)
      INTEGER TIME
C
C Output variables:
C Y       - contains the new value of the dependent variables.
C           Y(4): zone interior wall temperature (F)
C           Y(5): zone interior glass temperature (F)
C
C Procedures called: CNDCTI;CONVWA,GCNDCT,GCNVCT,GRADIA
C Common blocks: STATE1
C
      REAL                A
      LOGICAL                    L
      COMMON / STATE1 / A(100),L(100)
C
C Local variables:
C A41    - conduction component of heat transfer between interior of wall and
C          zone air.
C A42    - convection component of heat transfer between interior of wall and
C          zone air.
C A43    - radiation component of heat transfer between interior of wall and
C          zone air.
C A51    - conduction component of heat transfer between interior of wall and
C          outside air.
C A52    - convection component of heat transfer between interior of wall and
C          outside air.
C A53    - radiation component of heat transfer between interior of wall and
C          outside air.
C TI     -  zone dry bulb temperature (F)
C TOUTA  -  outside air dry bulb temperature (F)
C TRN    -  Mean radiant temperature in the zone (F)
C TW     -  zone internal wall temperature at mass center (F)
C
      EQUIVALENCE ( A(1),TOUTA )
C
      TI = Y(1)
      TW = Y(3)
      TRN = MRT(TIME)
C----------wall interior surface temperature  thermal energy balance ---
      A41 = CNDCTI(TIME)
      A42 = CONVWA(TIME)
      A43 = WRADIA(TIME)
      Y(4) = ( A41 * TW  +  A42 * TI + A43 * TRN +  WSOLAR(TIME) ) /
     &        ( A41 + A42 + A43 )
C----------glass interior surface temperature thermal energy balance ---
      A51 = GCNDCT(TIME)
      A52 = GCNVCT(TIME)
      A53 = GRADIA(TIME)
      IF(A51.NE.0.AND.A52.NE.0.AND.A53.NE.0)
     &Y(5) = ( A51 * TOUTA  +  A52 * TI  +  A53 * TRN ) /
     &        ( A51 + A52 + A53 )
C
      RETURN
```

```
      END
C=======================================================================
      FUNCTION ZONEFN(I,TIME,NSTVAR,Y)
C=======================================================================
C This function returns the right hand side of the linear ordinary
C differential equation for the building zone model. This should be a
C linear function of time, the set of dependent variables, boundary
C conditions, and parameters.
C
C Input variables:
C I      - Index of differential equation for which function is being
C          determined.
C TIME   - A time value which is a combination of the current minutes and
C          seconds of the time of day. Used to determine if time-dependent
C          functions should be recalculated
C NSTVAR - Total number of variables in the zone model state vector.
C Y      - Zone model state vector which is a subset of the simulation
C          state vector.
C
      INTEGER TIME
      REAL Y(NSTVAR)
C
C Output variables:
C ZONEFN - right hand side of linear ODE.
C
C Procedures called: CNDCTI,CNDCTO,CONVGL,CONVWA,HUMRAT,INFILT,INFILW,
C                    INTERN, MOIST,MASSA,MASSTH,MASSW,SOLAR,SUPPLW,
C                    SUPPLY
C
      REAL CNDCTI,CNDCTO,CONVGL,CONVWA,HUMRAT,INFILT,INFILW,INTERN,
     &     MOIST,MASSA,MASSTH,MASSW,SOLAR,SUPPLW,SUPPLY
C
C Common blocks: STATE1
C
      REAL               A
      LOGICAL                   L
      COMMON / STATE1 / A(100),L(100)
C
C Local Variables:
C MAXW   - Humidity ratio in the space at current temperature and 100% RH.
C PRESS  - Atmospheric pressure (assummed constant) in inches of Hg.
C TI     - zone dry bulb temperature (F)
C TIG    - zone interior glass temperature (F)
C TIW    - zone interior wall temperature (F)
C TOUTA  - TEMPERATURE of outside air (F)
C TW     - zone internal wall temperature at mass center (F)
C WI     - zone humidity ratio
C WOUTA  - HUMIDITY RATIO of outside air (lbw/lbda)
C
      PARAMETER (PRESS = 29.921)
      REAL MAXW
      EQUIVALENCE ( A(1),TOUTA ),( A(2),WOUTA )
C
C---------- Zone Air Energy Balance ------------------------------------
C
      IF(I.EQ.1)THEN
        TI = Y(1)
        WI = Y(2)
```

```
         TIW = Y(4)
         TIG = Y(5)
         ZONEFN = (SUPPLY(TIME) +
     &             INFILT(TIME,TI,WI) * (TOUTA  - TI) +
     &             CONVWA(TIME) *       (TIW    - TI) +
     &             CONVGL(TIME) *       (TIG    - TI) +
     &             INTERN(TIME))/ MASSTH(TIME,WI,TI)
C
C---------- Zone Air Moisture Balance -----------------------------------
C
      ELSE IF(I.EQ.2)THEN
         WI = Y(2)
         TI = Y(1)
         ZONEFN = (INFILW(TIME,TI) * (WOUTA - WI) +
     &             SUPPLW(TIME) +
     &             MOIST(TIME))/ MASSA(TIME,TI)
C
C If zone air reaches saturation, prevent further increase
C
         MAXW = HUMRAT(TI,100.0,PRESS)
         IF(WI.GT.MAXW.AND.ZONEFN.GT.0.0) ZONEFN = 0.0
         IF(WI.LE.0.0.AND.ZONEFN.LT.0.0) ZONEFN = 0.0
C
C---------- Wall Thermal Energy Balance ---------------------------------
C
      ELSE IF(I.EQ.3)THEN
         TW = Y(3)
         TIW = Y(4)
C
C NOTE: TOUTA in the equation below should be the solair temperature......
C
      ZONEFN = (CNDCTO(TIME) * (TOUTA  - TW) +
     &          CNDCTI(TIME) * (TIW    - TW) +
     &          SOLAR(TIME) )/ MASSW(TIME)
      ELSE
         ZONEFN = 0.
      ENDIF
C
C
      RETURN
      END
C=======================================================================
      FUNCTION SUPPLY(TIME)
C=======================================================================
C Returns the amount of energy entering or being removed from the zone
C by the HVAC system. This includes energy from the air handling unit
C and the local conditioning equipment.
C
C UNITS: returns Btu/sec.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
      INTEGER TIME
C
C Output variables:
C SUPPLY - Energy entering or being removed from the zone by the HVAC system.
C
```

```
      REAL SUPPLY
C
C Procedures called: ENTHAL
C Common blocks: STATE1
C
      REAL              A
      LOGICAL                  L
      COMMON / STATE1 / A(100),L(100)
C
C Local Variables:
C FUNC   - Holding variable for function return value.
C HSA1   - enthalpy of the supply air in BTU/lbm
C HSA2   - enthalpy of the hot deck (if existing) supply air Btu/lbm
C HZA    - enthalpy of zone air in Btu/lbm
C MRA    - mass flow rate of return air (lbm/s)
C MSA1   - mass flow rate of supply air in LBm/sec.
C MSA2   - mass flow rate of hot deck (if existing) supply air in LBm/sec.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C PRESS  - Atmospheric pressure (assummed constant) in inches of Hg.
C QLOCH  - energy added to the zone by local equipment during the time step.
C QLOCC  - energy taken from the zone by local equipment during the time step.
C TIMEDP - logical parameter, true if function is time dependent.
C TZA    - temperature of zone air in degrees F.
C WZA    - humidity ratio of zone air in lbm water/ lbm dry air.
C
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .TRUE.,PRESS = 29.921)
      INTEGER OLD
      DATA NOINIT/.TRUE./,OLD/-1/
C
      REAL MRA,MSA1,MSA2
      EQUIVALENCE ( A(06),TZA    ),( A(07),WZA    ),( A(33),HSA1   )
      EQUIVALENCE ( A(51),MRA    ),( A(62),QLOCH  ),( A(63),QLOCC  )
      EQUIVALENCE ( A(52),MSA1   )
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-----------------------------------------------------------------------
C       THE FOLLOWING VALUES ARE FOR DUAL DECK (NOT IMPLEMENTED)
        MSA2 = 0.0
        HSA2 = 0.0
C
        HZA = ENTHAL(TZA,WZA,PRESS)
        FUNC = (HSA1*MSA1) + (HSA2*MSA2) - (HZA*MRA) + QLOCH - QLOCC
C-----------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      SUPPLY = FUNC
      RETURN
      END
C=======================================================================
      FUNCTION INFILT(TIME,TI,WI)
C=======================================================================
C Returns the infiltration coefficient for energy gain/loss due to
C outside air infiltration.
C
C UNITS: returns Btu/sec F.
```

```
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C TI     - Dry bulb temperature of zone air in F.
C WI     - Humidity ratio of zone air in lbm water/lbm dry air.
C
      INTEGER TIME
C
C Output variables:
C INFILT - the infiltration coefficient for energy gain/loss.
C
      REAL INFILT
C
C Procedures called: AXRATE,CPAIR,RHOAIR
C Common blocks: STATE1,SHELL
C
      REAL                 A
      LOGICAL                    L
      COMMON / STATE1 / A(100),L(100)
C
      EQUIVALENCE ( A(1),TOUTA ),( A(2),WOUTA )
C
      REAL                               LENOW,LENIW,KOW,KIW
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
C CPIA is specific heat of infiltration air (average of inside and outside)
C FUNC   - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C RHOIA  - density of infiltration air (average of inside and outside)
C TIMEDP - logical parameter, true if function is time dependent.
C TOUTA  - the outdoor air temperature in degrees F.
C WOUTA  - the outdoor air humidity ratio in lbm water/lbm dry air.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .TRUE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-------------------------------------------------------------------
C
C  RHOAIR is a function returning the density of moist air
C  AXRATE is a function for the air exchange rate in air changes per second
C  VOLAIR is the volume of the zone air
C
      RHOIA = (RHOAIR(WOUTA,TOUTA) + RHOAIR(WI,TI)) / 2.
      CPIA = (CPAIR(WOUTA) + CPAIR(WI)) / 2.
      FUNC = CPIA * RHOIA * VOLAIR * AXRATE(TIME,TI)
C
C-------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      INFILT = FUNC
      RETURN
```

```
      END
C=====================================================================
      FUNCTION AXRATE(TIME,TI)
C=====================================================================
C Returns the air exchange rate for a building zone.
C
C UNITS: returns air exchange rate in air changes per second
C        TI and TO must be in degrees F
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C TI     - Dry bulb temperature of zone air in F.
C
      INTEGER TIME
C
C Output variables:
C INFILT - the infiltration coefficient for energy gain/loss.
C
      REAL AXRATE
C
C Procedures called: none
C Common blocks: STATE1,SHELL
C
      REAL                A
      LOGICAL                    L
      COMMON / STATE1 / A(100),L(100)
C
      EQUIVALENCE ( A(1),TOUTA ),( A(04),VWIND )
C
      REAL                                  LENOW,LENIW,KOW,KIW
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
C FUNC   - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C TOUTA  - the outdoor air temperature in degrees F.
C VWIND  - the outdoor wind velocity in miles per hour.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .TRUE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-------------------------------------------------------------------
C
C  AIREXC is the standard air exchange rate without the effects of wind
C         and temperature differential in air changes per hour:
C         = 1.5 for leaky  building
C         = 1.0 for standard building
C         = 0.5 for modern type building
C
      AXRATE=AIREXC/0.695*(0.15+0.013*VWIND+0.005*ABS(TOUTA-TI))
      FUNC = AXRATE / 3600.
C-------------------------------------------------------------------
```

```
            OLD = TIME
            NOINIT = .FALSE.
          ENDIF
          AXRATE = FUNC
          RETURN
          END
C=========================================================================
      FUNCTION CONVWA(TIME)
C=========================================================================
C Returns the convection coefficient for convection heat transfer between
C the zone air and the wall (not including glass) surfaces enclosing the
C zone.
C
C UNITS: returns Btu/s F.
C
C Input variables:
C TIME    - Used to determine if time-dependent functions should be recalculated.
C
      INTEGER TIME
C
C Output variables:
C CONVWA - the convection coefficient between zone air and wall.
C
      REAL CONVWA
C
C Procedures called: none
C Common blocks: SHELL
C
      REAL                                  LENOW,LENIW,KOW,KIW
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
C FUNC    - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD     - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .FALSE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-------------------------------------------------------------------------
C     HIWALL is the interior wall surface film coefficient
C     AREAW is the surface area of the wall exposed to the outside
C
      FUNC = HIWALL * AREAW
C-------------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      CONVWA = FUNC
      RETURN
      END
C=========================================================================
      FUNCTION CONVGL(TIME)
```

```
C=========================================================================
C Returns the convection coefficient for convection heat transfer between
C the zone air and the glass (windows) surfaces enclosing the zone.
C
C UNITS: returns Btu/s F.
C
      REAL CONVGL
C
C THIS FUNCTION NOT IMPLEMENTED (NO GLASS IN WALL).
C
      CONVGL = 0.0
      RETURN
      END
C=========================================================================
      FUNCTION INTERN(TIME)
C=========================================================================
C Returns the sensible internal gain in the zone from lighting, people and
C equipment.
C
C UNITS: returns Btu/s.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
C Output variables:
C INTERN - sensible internal gain.
C
      REAL INTERN
C
C Procedures called: SCHED
C Common blocks: none
C
      INTERN = SCHED(TIME,'S')
      RETURN
      END
C=========================================================================
      FUNCTION SCHED(TIME,SORL)
C=========================================================================
C This routine is used to determine the current values of sensible or
C latent internal gains in the zone. The values of the gains are determined
C from a schedule of occupancy , lighting and equipment usage.
C
C Input variables:
C SORL   - Character used to select the return of latent gain if 'L' and
C          sensible gain if 'S'.
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
      INTEGER TIME
      CHARACTER*1 SORL
C
C Output variables: SCHED
C SCHED  - Contains either sensible internal gain in Btu/sec. or latent
C          moisture gain in lbm water/sec.
C
C Procedures called: none
C Common blocks: STATE1,TIME,USE
C
      PARAMETER (NSCHED = 10)
```

```
      REAL                          TSCHED,          SENSIB,          LATENT
      COMMON / USE / LSCHED,TSCHED(NSCHED),SENSIB(NSCHED),LATENT(NSCHED)
     &                                     ,PESENS(NSCHED),PELATE(NSCHED)
C
      INTEGER           DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME   / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
C
      REAL               A
      LOGICAL                   L
      COMMON / STATE1 / A(100),L(100)
C
C Local Varlablas:
C
C DHOUR  - current hour of the day in dacimal hours.
C ISCHED - Index into schedule tables for sensible and latent loads for
C          current time of day.
C NOINIT - logical variable, true if function has never been calculated.
C OCCUPY - STATUS zone occupied (.true.)/unoccupied (.false.).
C OLD    - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT,OCCUPY
      PARAMETER (TIMEDP = .TRUE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      EQUIVALENCE ( L(11),OCCUPY )
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-------------------------------------------------------------------
C Find row of schedule table for current time of day
C
        DHOUR = HOUR + MINUTE/60. + SECOND/3600.
        DO 1000 I = 1,LSCHED
          IF(TSCHED(I).GT.DHOUR) GO TO 2000
 1000   CONTINUE
 2000   ISCHED = I - 1
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
C
C-------------------------------------------------------------------
C Determine sensible or latent load from tables.
C The following  assumes that the sensible and moisture gains of people
C are constant under all conditions. This is not strictly correct.
C
      IF(SORL.EQ.'L')THEN
        SCHED = LATENT(ISCHED) + PELATE(ISCHED)
      ELSE IF(SORL.EQ.'S')THEN
        SCHED = SENSIB(ISCHED) + PESENS(ISCHED)
C
C-------------------------------------------------------------------
C determine if building is occupied
C
        IF(PESENS(ISCHED).GT.0)THEN
          OCCUPY = .TRUE.
        ELSE
          OCCUPY = .FALSE.
```

```
          ENDIF
        ELSE
          SCHED = 0.
        ENDIF
        RETURN
        END
C=======================================================================
        FUNCTION MASSTH(TIME,WI,TI)
C=======================================================================
C Returns the thermal mass of the zone air and furnishings (not including
C walls).
C
C UNITS: returns Btu/F.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C TI     - Dry bulb temperature of zone air in F.
C WI     - Humidity ratio of zone air in lbm water/lbm dry air.
C
        INTEGER TIME
C
C Output variables:
C MASSTH - the thermal mass of the zone air and furnishings in Btu/F.
C
        REAL MASSTH
C
C Procedures called: CPAIR,RHOAIR
C Common blocks: SHELL
C
        REAL                                LENOW,LENIW,KOW,KIW
        COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                   VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
C CAIR   - thermal mass of the zone air.
C FUNC   - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C
        INTEGER OLD
        LOGICAL TIMEDP,NOINIT
        PARAMETER (TIMEDP = .TRUE.)
        DATA NOINIT/.TRUE./,OLD/-1/
C
        IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C----------------------------------------------------------------------
C       VOLAIR is the volume of the zone air
C       RHOAIR is a function returning the density of moist air
C       CPAIR is a function returning the specific heat of moist air
C       CFUR is the thermal mass of the zone furnishings
C
          CAIR = CPAIR(WI) * RHOAIR(WI,TI) * VOLAIR
          FUNC = CFUR + CAIR
C----------------------------------------------------------------------
          OLD = TIME
          NOINIT = .FALSE.
        ENDIF
```

```
      MASSTH = FUNC
      RETURN
      END
C=========================================================================
      FUNCTION INFILW(TIME,TI,WI)
C=========================================================================
C Returns the infiltration coefficient for moisture gain/loss by outside
C air infiltration.
C
C UNITS: returns lbm dry air/second.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C TI     - Dry bulb temperature of zone air in F.
C WI     - Humidity ratio of zone air in lbm water/lbm dry air.
C
      INTEGER TIME
C
C Output variables:
C INFILW - the mass flow rate of dry air from outside to inside which
C          carries water vapor with it. The actual change in moisture will
C          be assumed to be: INFILW times the indoor-outdoor humidity ratio
C          differential.
C
      REAL INFILW
C
C Procedures called: AXRATE,RHOAIR
C Common blocks: SHELL,STATE1
C
      REAL               A
      LOGICAL                    L
      COMMON / STATE1 / A(100),L(100)
C
      REAL                                     LENOW,LENIW,KOW,KIW
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
C FUNC   - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C RHOIA  - density of DRY infiltration air (average of inside and outside)
C TIMEDP - logical parameter, true if function is time dependent.
C TOUTA  - the outdoor air temperature in degrees F.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .TRUE.)
      EQUIVALENCE ( A(1),TOUTA )
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-----------------------------------------------------------------------
C
C  RHOAIR is a function returning the density of moist air
C  AXRATE is a function for the air exchange rate in air changes per second
C  VOLAIR is the volume of the zone air.
C
```

```
      RHOIA = (RHOAIR(0.0,TOUTA) + RHOAIR(0.0,TI)) / 2.
      FUNC = RHOIA * VOLAIR * AXRATE(TIME,TI)
C-----------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      INFILW = FUNC
      RETURN
      END
C=======================================================================
      FUNCTION SUPPLW(TIME)
C=======================================================================
C Returns the net amount of water entering the zone from the air handling
C unit. If negative, water is being extracted from the zone.
C
C UNITS: lbm water/ second.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
      INTEGER TIME
C
C Output variables:
C SUPPLW - water entering or being removed from the zone by the HVAC system.
C
      REAL SUPPLW
C
C Procedures called: none
C Common blocks: STATE1
C
      REAL              A
      LOGICAL                L
      COMMON / STATE1 / A(100),L(100)
C
C Local Variables:
C FUNC   - Holding variable for function return value.
C MRA    - mass flow rate of return air (lbm/s)
C MSA1   - mass flow rate of supply air in LBm/sec.
C MSA2   - mass flow rate of hot deck (if existing) supply air in LBm/sec.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C PRESS  - Atmospheric pressure (assummed constant) in inches of Hg.
C TIMEDP - logical parameter, true if function is time dependent.
C WRA    - humidity ratio of return air in lbm water/ lbm dry air.
C WSA1   - humidity ratio of supply air in lbm water/ lbm dry air.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .TRUE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      REAL MRA,MSA1,MSA2,WRA,WSA1,WSA2
      EQUIVALENCE ( A(22),WRA    ),( A(51),MRA    )
      EQUIVALENCE ( A(34),WSA1   ),( A(52),MSA1   )

C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-----------------------------------------------------------------------
```

```
C       THE FOLLOWING VALUES ARE FOR DUAL DECK (NOT IMPLEMENTED)
        MSA2 = 0.0
        WSA2 = 0.0
C
C NOTE: this is approximate, since MRA and MSA1 are in lb/sec of moist air.
C
        FUNC = (WSA1 * MSA1) + (WSA2 * MSA2) - (WRA * MRA)
C-----------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      SUPPLW = FUNC
      RETURN
      END
C=======================================================================
      FUNCTION MOIST(TIME)
C=======================================================================
C Returns the amount of water generated in the zone by people and equipment.
C
C UNITS: lbm water/ second.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
C Output variables:
C MOIST - moisture gain from people and equipment.
C
      REAL MOIST
C
C Procedures called: SCHED
C Common blocks: none
C
      MOIST  =  SCHED(TIME,'L')
      RETURN
      END
C=======================================================================
      FUNCTION MASSA(TIME,TI)
C=======================================================================
C Returns the mass of the zone air which can absorb and release water
C vapor. (mass of dry air)
C
C UNITS: returns lbm of dry air.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C TI     - Dry bulb temperature of zone air in F.
C
      INTEGER TIME
C
C Output variables:
C MASSA  - the infiltration coefficient for energy gain/loss.
C
      REAL MASSA
C
C Procedures called: RHOAIR
C Common blocks: SHELL
C
      REAL                                  LENOW,LENIW,KOW,KIW
```

```
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
C FUNC   - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C WI     - zone humidity ratio, set to zero for dry air.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .TRUE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-----------------------------------------------------------------------
C     VOLAIR is the volume of the zone air.
C     RHOAIR is a function returning the density of moist air.
C     RMGFUR is a factor used to produce the effective air mass for moisture
C            absorption by the zone furnishings and air.
C
        WI = 0.0
        FUNC = RHOAIR(WI,TI) * VOLAIR * RMGFUR
C-----------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      MASSA = FUNC
      RETURN
      END
C=======================================================================
      FUNCTION CNDCTO(TIME)
C=======================================================================
C Returns the coefficient for thermal conductance between the mass center
C of the wall and the environment outside of the building.
C
C UNITS: returns Btu/second F.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
      INTEGER TIME
C
C Output variables:
C CNDCTO - conductance coefficient between outside air and wall mass center.
C
      REAL CNDCTO
C
C Procedures called: none
C Common blocks: SHELL
C
      REAL                                LENOW,LENIW,KOW,KIW
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
C FUNC   - Holding variable for function return value.
```

```
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .FALSE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-----------------------------------------------------------------------
C     HOWALL is the surface film coefficient for the outside wall
C     LENOW is the thickness  of the wall outside of the mass center
C     KOW is the thermal conductivity of the wall outside of the mass
C        center
C     AREAW is the area of the wall exposed to the outside
C
      FUNC = 1 / (1/HOWALL + LENOW/KOW) * AREAW
C-----------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      CNDCTO = FUNC
      RETURN
      END
C=======================================================================
      FUNCTION CNDCTI(TIME)
C=======================================================================
C Returns the coefficient for thermal conductance between the mass center
C of the wall and the inner wall surface surrounding the zone.
C
C UNITS: returns Btu/second F.
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
      INTEGER TIME
C
C Output variables:
C CNDCTI - conductance coefficient between zone air and wall mass center.
C
      REAL CNDCTI
C
C Procedures called: none
C Common blocks: SHELL
C
      REAL                                 LENOW,LENIW,KOW,KIW
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMOFUR
C
C Local Variables:
C FUNC   - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
```

```
      PARAMETER (TIMEDP = .FALSE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-----------------------------------------------------------------------
C        KIW is the thermal conductivity of the wall inside of the mass
C            center
C        LENIW is the thickness of the wall inside the mass center
C        AREAW is the surface area of wall exposed to the outside
C
        FUNC = KIW / LENIW * AREAW
C-----------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      CNDCTI = FUNC
      RETURN
      END
C=======================================================================
      FUNCTION SOLAR(TIME)
C=======================================================================
C Returns the amount of solar heat gain through the windows in the zone.
C
C UNITS: returns Btu/second.
C
      REAL SOLAR
C
C
C THIS FUNCTION NOT IMPLEMENTED (NO GLASS IN WALL).
C
      SOLAR = 0.0
      RETURN
      END
C=======================================================================
      FUNCTION MASSW(TIME)
C=======================================================================
C Returns the thermal capacitance of the wall surrounding the zone.
C
C UNITS: returns Btu/F
C
C Input variables:
C TIME   - Used to determine if time-dependent functions should be recalculated.
C
      INTEGER TIME
C
C Output variables:
C MASSW  - thermal capacitance of wall.
C
      REAL MASSW
C
C Procedures called: none
C Common blocks: SHELL
C
      REAL                                  LENOW,LENIW,KOW,KIW
      COMMON / SHELL / AREAW,HIWALL,HOWALL,LENOW,LENIW,KOW,KIW,CFUR,
     &                 VOLAIR,CPOW,RHOOW,CPIW,RHOIW,AIREXC,RMGFUR
C
C Local Variables:
```

```
C FUNC   - Holding variable for function return value.
C NOINIT - logical variable, true if function has never been calculated.
C OLD    - time when function was last calculated.
C TIMEDP - logical parameter, true if function is time dependent.
C
      INTEGER OLD
      LOGICAL TIMEDP,NOINIT
      PARAMETER (TIMEDP = .FALSE.)
      DATA NOINIT/.TRUE./,OLD/-1/
C
      IF(NOINIT.OR.(TIME.NE.OLD.AND.TIMEDP)) THEN
C-----------------------------------------------------------------------
C   CPOW is the specific heat of the wall outside of mass center
C   RHOOW is the density of the wall outside of mass center.
C   LENOW is the thickness of the wall outside of mass center.
C   CPIW is the specific heat of the wall inside of mass center
C   RHOIW is the density of the wall inside of mass center.
C   LENIW is the thickness of the wall inside of mass center.
C   AREAW is the surface area of wall exposed to the outside
C
        FUNC = (CPOW * RHOOW * LENOW  +  CPIW * RHOIW * LENIW) * AREAW
C-----------------------------------------------------------------------
        OLD = TIME
        NOINIT = .FALSE.
      ENDIF
      MASSW = FUNC
      RETURN
      END
C=======================================================================
      FUNCTION WRADIA(TIME)
C=======================================================================
C Returns the coefficient for radiation heat transfer between the wall and
C any other walls or furnishings (at room mean radiant temperature).
C
      REAL WRADIA
C
C THIS FUNCTION NOT IMPLEMENTED (no radiation assumed).
C
      WRADIA = 0.0
      RETURN
      END
C=======================================================================
      FUNCTION WSOLAR(TIME)
C=======================================================================
C Returns the amount of solar energy entering the zone which is absorbed
C by the wall surface.
C
      REAL WSOLAR
C
C THIS FUNCTION NOT IMPLEMENTED (no windows assumed).
C
      WSOLAR = 0.0
      RETURN
      END
C=======================================================================
      FUNCTION GCNDCT(TIME)
C=======================================================================
C Returns the coefficient for conductance through the glass in the zone
```

```
C from the inner wall surface and the environment outside the building.
C
      REAL GCNDCT
C
C
C THIS FUNCTION NOT IMPLEMENTED (no windows assumed).
C
      GCNDCT = 0.0
      RETURN
      END
C=========================================================================
      FUNCTION GCNVCT(TIME)
C=========================================================================
C Returns the coefficient for convection between the inner glass surface
C and the air in the zone.
C
      REAL GCNVCT
C
C
C THIS FUNCTION NOT IMPLEMENTED (no windows assumed).
C
      GCNVCT = 0.0
      RETURN
      END
C=========================================================================
      FUNCTION GRADIA(TIME)
C=========================================================================
C Returns the coefficient for radiation heat transfer between the glass
C and any other surfaces or furnishings in the zone.
C
      REAL GRADIA
C
C
C THIS FUNCTION NOT IMPLEMENTED (no windows assumed).
C
      GRADIA = 0.0
      RETURN
      END
C=========================================================================
      FUNCTION MRT(TIME)
C=========================================================================
C Returns the value of the mean radiant temperature in the zone.
C
      REAL MRT
C
C
C THIS FUNCTION NOT IMPLEMENTED (no radiation transfer assumed).
C
      MRT = 0.0
      RETURN
      END
C=========================================================================
      SUBROUTINE COMFRT(TIMEST)
C=========================================================================
C This routine is used to determine if a zone space is within the limits
C of comfort. Comfort is defined in accordance with ASHRAE standard
C 55-74 as being acceptable when the dry bulb temperature and humidity
C ratio are within certain limits. The check of comfort is only performed
```

```
C If the building zone is occupied, as determined by the logical variable
C OCCUPY in the state vector.
C
C Input variables:
C TIMEST - The current value to be used for the major simulation time step.
C
      INTEGER TIMEST
C
C Output variables: none
C Procedures called: RELHUM
C Common blocks: STATE1
C
      REAL                A
      LOGICAL                   L
      COMMON / STATE1 / A(100),L(100)
C
C Local Variables:
C OCCUPY - STATUS zone occupied (.true.)/unoccupied (.false.).
C OUT    - Logical variable which is true when comfort is outside of limits.
C PRESS  - Atmospheric pressura (assummed constant) in inches of Hg.
C RHAMAX - COMFORT relative humidity at maximum temperature (occupied) (%)
C RHAMIN - COMFORT relative humidity at minimum temperature (occupied) (%)
C SECOUT - COMFORT seconds outside of comfort range
C TCLEFT - Minimum temperature comfort limit, function of humidity ratio.
C TCRITE - Maximum temperature comfort limit, function of humidity ratio.
C TZA    - temperature of zone air in degrees F.
C TZMAX  - COMFORT maximum dry bulb temperature during occupied period (F).
C TZMIN  - COMFORT minimum dry bulb temperature during occupied period (F).
C WZA    - humidity ratio of zone air in lbm water/ lbm dry air.
C
      PARAMETER (PRESS = 29.921)
      LOGICAL OCCUPY,OUT
      EQUIVALENCE ( L(11),OCCUPY ) , ( A(06),TZA    ) , ( A(07),WZA   )
      EQUIVALENCE ( A(90),SECOUT ) , ( A(91),TZMAX  ) , ( A(92),TZMIN )
      EQUIVALENCE ( A(93),RHAMAX ) , ( A(94),RHAMIN )
C
C-----------------------------------------------------------------------
C Test for conditions outside of comfort limits
C
      IF(OCCUPY) THEN
        TCLEFT = 72.58 - 131.58*(WZA)
        TCRITE = 81.73 - 394.74*(WZA)
        IF(WZA.GT.0.012.OR.WZA.LT.0.0044) THEN
          OUT = .TRUE.
        ELSE IF(TZA.LT.TCLEFT.OR.TZA.GT.TCRITE) THEN
          OUT = .TRUE.
        ELSE
          OUT = .FALSE.
        ENDIF
        IF(OUT) SECOUT = SECOUT + TIMEST
C
C-----------------------------------------------------------------------
C Test for max and min space conditions
C
        IF(TZA.GT.TZMAX) THEN
          TZMAX = TZA
          RHAMAX = RELHUM(TZA,WZA,PRESS)
        ELSE IF(TZA.LT.TZMIN)THEN
```

```
              TZMIN = TZA
              RHAMIN = RELHUM(TZA,WZA,PRESS)
            ENDIF
          ENDIF
C
C----------------------------------------------------------------------
C Test for max rate of temperature change
C
C NOT IMPLEMENTED
C
      RETURN
      END
C======================================================================
      SUBROUTINE CMPILE(TIMEST)
C======================================================================
C This routine is used to compile cumulative totals of energy use.
C Four types of energy use are compiled. These are:
C 1. Fan Energy Use
C 2. Economizer Energy = energy provided by economizer.
C 3. Load Energy = energy required to heat or cool air to space entering
C                  condition.
C 4. Reheat/recool Energy = energy used to reheat and recool air for space
C                           temperature control.
C Total Thermal Energy Use is also compiled. All thermal energy values have
C a heating component and a cooling component.
C
C Input variables:
C TIMEST - The current value to be used for the major simulation time step.
C
      INTEGER TIMEST
C
C Output variables: none
C Procedures called: RHOAIR,SHOTS
C Common blocks: HVAC,STATE1

      LOGICAL         DUDECK,HUMCON,RAFAN,VAV
      COMMON / HVAC / DUDECK,HUMCON,RAFAN,VAV,VOLSAF,CAPLOC,PGAINL,
     &                TCTHSE,QRAFAN,FVAMIN,QSAFAN,HCAP1,CCAP1,DTSETB,
     &                POWSAF,POWRAF,GAINIL
C
      REAL              A
      LOGICAL              L
      COMMON / STATE1 / A(100),L(100)
C
C Local Variables:
C EACTC  - actual energy for cooling (Btu)
C EACTH  - actual energy for heating (Btu)
C EECOC  - COOLING ENERGY (economizer) (Btu)
C EECOH  - HEATING ENERGY (economizer) (Btu)
C EFANS  - ELECTRICAL ENERGY to fans (kWh)
C ELOADC - COOLING ENERGY (load) (Btu)
C ELOADH - HEATING ENERGY (load) (Btu)
C ELOCC  - COOLING ENERGY from local zone equipment (Btu)
C ELOCH  - HEATING ENERGY to local zone equipment (Btu)
C EREC   - COOLING ENERGY (recool) (Btu)
C EREH   - HEATING ENERGY (reheat) (Btu)
C EREQC  - COOLING ENERGY (requirements) (Btu)
C EREQH  - HEATING ENERGY (requirements) (Btu)
```

```
C HMA    - ENTHALPY of mixed air (BTU/lb)
C HMC    - ENTHALPY of return air plus ventilation air(BTU/lb)
C HSA1   - ENTHALPY of supply air (BTU/lb)                           cold deck
C HSA2   - ENTHALPY of hot deck (if existing) supply air in (Btu/lbm)
C MRA    - mass flow rate of return air (lbm/s)
C MSA1   - mass flow rate of supply air in LBm/sec.
C MSA2   - mass flow rate of hot deck (if existing) supply air in LBm/sec.
C POWR   - electric power to return air fan (kW)
C POWS   - electric power to supply air fan (kW)
C QACT1  - actual energy for cold deck (or single deck) (Btu).
C QACT2  - actual energy for hot deck (if existing) (Btu).
C QLOAD  - load energy (Btu), may be negative for energy extraction (cooling)
C QREQ1  - requirements energy for cold deck (or single deck) (Btu).
C QREQ2  - requirements energy for hot deck (if existing) (Btu).
C RVOLR  - ratio of current return volume flow rate to rated volume flow rate
C RVOLS  - ratio of current supply volume flow rate to rated volume flow rate
C TRA    - TEMPERATURE of return air (F)
C TSA1   - TEMPERATURE of supply air (F)                             cold deck
C TSA2   - Temperature of hot deck (if existing) supply air (F)
C VRA    - Volume flow rate of return air in ft^3/sec.
C VSA    - Volume flow rate of supply air in ft^3/sec.
C WRA    - HUMIDITY RATIO of return air (lbw/lbda)
C WSA1   - HUMIDITY RATIO of supply air (lbw/lbda)                   cold deck
C WSA2   - Humidity ratio of hot deck (if existing) supply air (lbw/lbda)
C
      REAL MSA1,MSA2,MRA
      EQUIVALENCE ( A(52),MSA1   ),( A(33),HSA1   ),( A(24),HMA    )
      EQUIVALENCE ( A(64),EREQH  ),( A(65),EREQC  ),( A(35),HMC    )
      EQUIVALENCE ( A(66),EECOH  ),( A(67),EECOC  ),( A(68),ELOADH )
      EQUIVALENCE ( A(69),ELOADC ),( A(73),ELOCH  ),( A(70),EREH   )
      EQUIVALENCE ( A(71),EREC   ),( A(72),EFANS  ),( A(34),WSA1   )
      EQUIVALENCE ( A(32),TSA1   ),( A(20),TRA    ),( A(22),WRA    )
      EQUIVALENCE ( A(51),MRA    ),( A(74),ELOCC  )
C
C-----------------------------------------------------------------
C Fan Energy
C
C These calculations are based on fan law number 3 from ASHRAE equipment
C handbook 1983, page 3.5, table 2.
C
      MSA2 = 0.
      WSA2 = 0.
      TSA2 = 0.
      IF(MSA1.GT.0.0) THEN
        VSA = MSA1 / RHOAIR(WSA1,TSA1)
        IF(MSA2.GT.0.0) VSA = VSA + MSA2 / RHOAIR(WSA2,TSA2)
        RVOLS = VSA / VOLSAF
        POWS = POWSAF * RVOLS * RVOLS * RVOLS
        EFANS = EFANS + POWS * TIMEST / 3600.
        IF(RAFAN) THEN
          VRA = MRA / RHOAIR(WRA,TRA)
          RVOLR = VRA / VOLSAF
          POWR = POWRAF * RVOLR * RVOLR * RVOLR
          EFANS = EFANS + POWR * TIMEST / 3600.
        ENDIF
      ENDIF
C
C-----------------------------------------------------------------
```

```
C Requirements Energy
C
      MSA2 = 0.0
      HSA2 = 0.0
C
      QREQ1 = MSA1 * (HSA1 - HMC) * TIMEST
      QREQ2 = MSA2 * (HSA2 - HMC) * TIMEST
      IF(QREQ1.GT.0.) THEN
        EREQH = EREQH + QREQ1
      ELSE
        EREQC = EREQC + QREQ1
      ENDIF
      IF(QREQ2.GT.0.) THEN
        EREQH = EREQH + QREQ2
      ELSE
        EREQC = EREQC + QREQ2
      ENDIF
C
C-----------------------------------------------------------------------
C Actual Energy
C
      QACT1 = MSA1 * (HSA1 - HMA) * TIMEST
      QACT2 = MSA2 * (HSA2 - HMA) * TIMEST
      IF(QACT1.GT.0.) THEN
        EACTH = EACTH + QACT1
      ELSE
        EACTC = EACTC + QACT1
      ENDIF
      IF(QACT2.GT.0.) THEN
        EACTH = EACTH + QACT2
      ELSE
        EACTC = EACTC + QACT2
      ENDIF
C
C-----------------------------------------------------------------------
C Economizer Energy
C
      EECØH = EREQH - EACTH
      EECØC = EREQC - EACTC
C
C
C-----------------------------------------------------------------------
C Load Energy
C
      QLØAD =( (MSA1*HSA1 + MSA2*HSA2) - (MSA1+MSA2)*HMA ) * TIMEST
      IF(QLØAD.GT.0.) THEN
        ELØADH = ELØADH + QLØAD
      ELSE
        ELØADC = ELØADC + QLØAD
      ENDIF
C
C-----------------------------------------------------------------------
C Reheat/Recool Energy
C
      EREH = EACTH - ELØADH + ELØCH
      EREC = EACTC - ELØADC + ELØCC
C
C***********************************************************************
```

```
      CALL SHOTS
C*********************************************************************
      RETURN
      END
C=====================================================================
      SUBROUTINE REPORT
C=====================================================================
C This routine is used to output the final energy usage numbers at the
C conclusion of a test.
C The output is current sent to the console but should eventually be
C transmitted to a file for further analysis by an addtional program.
C
C Input variables: none
C Output variables: none
C Procedures called: none
C Common blocks: PFILES,STATE1,UNITS
C
      INTEGER            CLU,FLU,ILU,PLU
      COMMON / UNITS  / CLU,FLU,ILU,PLU
C
      CHARACTER*15       HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
      COMMON / PFILES / HVFILE,USFILE,CLFILE,SHFILE,INFILE,SEASON
C
      REAL               A
      LOGICAL                  L
      COMMON / STATE1 / A(100),L(100)
C
C Local Variables:
C
C EECOC  - COOLING ENERGY (economizer) (Btu)
C EECOH  - HEATING ENERGY (economizer) (Btu)
C EFANS  - ELECTRICAL ENERGY to fans (kWh)
C ELOADC - COOLING ENERGY (load) (Btu)
C ELOADH - HEATING ENERGY (load) (Btu)
C EREC   - COOLING ENERGY (recool) (Btu)
C EREH   - HEATING ENERGY (reheat) (Btu)
C EREQC  - COOLING ENERGY (requirements) (Btu)
C EREQH  - HEATING ENERGY (requirements) (Btu)
C RHAMAX - COMFORT relative humidity at maximum temperature (occupied) (%)
C RHAMIN - COMFORT relative humidity at minimum temperature (occupied) (%)
C SECOUT - COMFORT seconds outside of comfort range
C TZMAX  - COMFORT maximum dry bulb temperature during occupied period (F).
C TZMIN  - COMFORT minimum dry bulb temperature during occupied period (F).
C
      EQUIVALENCE ( A(64),EREQH  ),( A(65),EREQC  ),( A(72),EFANS  )
      EQUIVALENCE ( A(66),EECOH  ),( A(67),EECOC  ),( A(68),ELOADH )
      EQUIVALENCE ( A(69),ELOADC ),( A(70),EREH   ),( A(71),EREC   )
      EQUIVALENCE ( A(90),SECOUT ) , ( A(91),TZMAX  ) , ( A(92),TZMIN )
      EQUIVALENCE ( A(93),RHAMAX ) , ( A(94),RHAMIN )
C
      WRITE(CLU,7) HVFILE
      WRITE(CLU,8) USFILE
      WRITE(CLU,9) CLFILE
      WRITE(CLU,10) SHFILE
      WRITE(CLU,11) INFILE
      WRITE(CLU,12) SEASON
C
      WRITE (CLU,1)
```

```
      WRITE (CLU,2) EFANS
      WRITE (CLU,1)
      WRITE (CLU,3)
      WRITE (CLU,1)
      WRITE (CLU,4) EECOH,-EECOC
      WRITE (CLU,1)
      WRITE (CLU,5) ELOADH,-ELOADC
      WRITE (CLU,1)
      WRITE (CLU,6) EREH,-EREC
C
      WRITE (CLU,13) SECOUT/60.
      WRITE (CLU,14) TZMAX,RHAMAX
      WRITE (CLU,15) TZMIN,RHAMIN
C
      RETURN
    1 FORMAT(/,1X,18X,4('-'),'+',2(16('-'),'+'))
    2 FORMAT(1X,'FAN ENERGY                |',G12.5,' KWH|')
    3 FORMAT(1X,22X,'|    HEATING      |      COOLING    |')
    4 FORMAT(1X,'ECONOMIZER ENERGY      |',2(G12.5,' BTU|'))
    5 FORMAT(1X,'LOAD ENERGY            |',2(G12.5,' BTU|'))
    6 FORMAT(1X,'REHEAT/RECOOL ENERGY   |',2(G12.5,' BTU|'))
    7 FORMAT(1X,'HVAC     PARAMETERS FROM:',A15)
    8 FORMAT(1X,'USE      PARAMETERS FROM:',A15)
    9 FORMAT(1X,'CLIMATE PARAMETERS FROM:',A15)
   10 FORMAT(1X,'SHELL   PARAMETERS FROM:',A15)
   11 FORMAT(1X,'INITIALIZING DATA  FROM:',A15)
   12 FORMAT(1X,'WEATHER SEASON     USED:',A15)
   13 FORMAT(/,1X,'MINUTES OUTSIDE COMFORT ENVELOPE = ',F10.2)
   14 FORMAT(1X,'MAXIMUM ZONE TEMPERATURE = ',F10.2,' (',F5.1,'%RH)')
   15 FORMAT(1X,'MINIMUM ZONE TEMPERATURE = ',F10.2,' (',F5.1,'%RH)')
      END
C=======================================================================
      SUBROUTINE SHOTS
C=======================================================================
C This is a special subroutine used for development of the emulator model
C which will eventually be eliminated. Its purpose is to allow snapshots
C of the state vector to be taken and stored in a file for analysis.
C Instructions on which state vector variables are to be shot and what
C interval the snapshots are to be taken at, are read from the file
C assigned to logical unit 7. The snapshots are stored in the file
C assigned to logical unit 10.
C
      INTEGER DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      COMMON / TIME / DAY,HOUR,MINUTE,SECOND,ENDDAY,ENDHR,ENDMIN,ENDSEC
      LOGICAL L
      COMMON / STATE1 / A(100),L(100)
C
      LOGICAL NOINIT
      INTEGER IVECP(21)
      DATA NOINIT /.TRUE./
C
      IF(NOINIT)THEN
        READ(7,*,ERR=900,END=900) REPINV,NCHAN,(IVECP(I),I=1,NCHAN)
        IF(NCHAN.GT.21)
     &                  STOP 'TOO MANY STATE VECTOR POSITIONS IN SNAPSHOT'
        GO TO 1000
  900   WRITE(1,FMT='(1X,"REPORT INTERVAL IN TIMESTEPS?")')
        READ(1,*) REPINV
```

```
            WRITE(1,FMT='(1X,"NUMBER OF VECTOR POSITIONS TO REPORT?")')
            READ(1,*) NCHAN
            WRITE(1,FMT='(1X,"POSITION INDICES?")')
            READ(1,*) (IVECP(I),I=1,NCHAN)
 1000       REPLEF = 1
            NOINIT = .FALSE.
          ENDIF
          REPLEF = REPLEF - 1
          IF(REPLEF.LE.0) THEN
            RTIME = FLOAT(DAY-1)*24. + HOUR +
     &       FLOAT(MINUTE)/60. + FLOAT(SECOND)/3600.
            WRITE(10,1)RTIME,(A(IVECP(I)),I=1,NCHAN)
    1 FORMAT(1X,G11.5,6G11.3:,/,(1X,7G11.3))
            REPLEF = REPINV
          ENDIF
          RETURN
          END
C Version 1.5 - FEBRUARY 4, 1985 - W.B. MAY ,NATIONAL BUREAU OF STANDARDS
C=======================================================================
          FUNCTION CPAIR(W)
C=======================================================================
C This function returns the specific heat of moist air as a function of
C the humidity ratio.
C
C UNITS: specific heat is returned in Btu/lbm F
C
C REFERENCE: ASHRAE Handbook of Fundamentals 1981, page 5.3
C
          CPAIR = 0.24 + 0.444 * W
C
          RETURN
          END
C=======================================================================
          FUNCTION RHOAIR(W,TF)
C=======================================================================
C This function returns the density of moist air as a function of
C the humidity ratio and the dry bulb temperature.
C
C UNITS: density is returned in lbm/ft^3
C        temperature TF must be in degrees F
C
C REFERENCE: ASHRAE Handbook of Fundamentals 1981, page 5.3
C
C PRESSR is atmospheric pressure in lbf/ft^2 (assumed standard atmosphere)
C RAIR is the gas constant for dry air in ft-lbf/lbm R
C TA is the absoluta temperature of the air in degrees R
C FTOR is the conversion factor from F to R degrees
C
          PARAMETER (PRESSR = 2116.2, RAIR = 53.352, FTOR = 459.67)
C
          TA = TF + FTOR
          RHOAIR = PRESSR / (RAIR * TA * (1 - 1.6078 * W) )
          RETURN
          END
C=======================================================================
          FUNCTION ENTHAL(TDB,WPPDA,PA)
C=======================================================================
C from ASHRAE 1981 Fundamentals handbook, Page 5.4
```

```
C
C TDB = average dry bulb temperature in degrees F
C PA  = atmospheric pressure in inches of mercury
C WPPDA = humidity ratio in pounds of water per pound of dry air
C ENTHAL = enthalpy of mixture in BTU per pound of dry air
C
      ENTHAL = 0.240 * TDB + WPPDA * (1061. + 0.444*TDB)
      RETURN
      END
C======================================================================
      FUNCTION TEMP(ENTHAL,WPPDA,PA)
C======================================================================
C Based on equation for anthalpy in ASHRAE 1981 Fundamentals
C handbook, Page 5.4, solved for TEMP.
C
C TEMP = average dry bulb temperature in degrees F
C PA  = atmospheric pressure in inches of mercury
C WPPDA = humidity ratio in pounds of water per pound of dry air
C ENTHAL = enthalpy of mixture in BTU per pound of dry air
C
      TEMP = (ENTHAL - 1061.*WPPDA) / (0.240  + 0.444*WPPDA)
      RETURN
      END
C======================================================================
      FUNCTION DEWPT(WPPDA,PA)
C======================================================================
C from ASHRAE 1981 Fundamentals handbook, Page 5.4, equation 40b
C equation is valid in the range of TDB from 32 to 150 F.
C
C PA  = atmospheric pressure in inches of mercury
C WPPDA = humidity ratio in pounds of water per pound of dry air
C DEWPT = dewpoint temperature of mixture in degrees F.
C PW = water vapor partial pressure for the mixture in inches Hg.
C
      PW = PA*WPPDA / (0.62198 + WPPDA)
      ALPHA = ALOG(PW)
      DEWPT = 79.047 + 30.5790*ALPHA + 1.8893*ALPHA*ALPHA
      RETURN
      END
C======================================================================
      FUNCTION HUMRAT(TDB,RH,PA)
C======================================================================
C from ASHRAE 1981 Fundamentals handbook, Page 5.4
C
C TDB = average dry bulb temperature in degrees F
C PA  = atmospheric pressure in inches of mercury
C RH = relative humidity in percent
C HUMRAT = humidity ratio in pounds of water per pound of dry air
C
      PASCAL = PA * 3.38638E+3
      TDBK = (((TDB-32)/1.8)+273.15)
      PWAT = RH * PWSAT(TDBK) / 100.
      HUMRAT  = 0.62198 * ( PWAT / (PASCAL - PWAT) )
      RETURN
      END
C======================================================================
      FUNCTION RELHUM(TDB,WAIR,PA)
C======================================================================
```

```
      C from ASHRAE 1981 Fundamentals handbook, Page 5.4
      C
      C TDB = average dry bulb temperature in degrees F
      C PA  = atmospheric pressure in inches of mercury
      C RELHUM = relative humidity in percent
      C WAIR = humidity ratio in pounds of water per pound of dry air
      C
            PASCAL = PA * 3.38638E+3
            TDBK = (((TDB-32)/1.8)+273.15)
            PWAT = PASCAL * WAIR / (.62198 + WAIR)
            RELHUM = PWAT / PWSAT(TDBK) * 100.
            RETURN
            END
      C=======================================================================
            FUNCTION PWSAT(T)
      C=======================================================================
      C from ASHRAE 1981 Fundamentals handbook, Page 5.2
      C
      C TDB = dry bulb temperature in degrees K.
      C PWSAT = saturation pressure of water vapor in pascals
      C
            DATA C1/-5674.5359      /, C2/6.3925247    /, C3/-0.9677843E-2 /
            DATA C4/    0.62215701E-6/,C5/0.20747825E-8/, C6/-0.9484024E-12/
            DATA C7/    4.1635019   /, C8/-5800.2206   /, C9/ 1.3914993    /
            DATA C10/  -0.04860239  /,C11/0.41764768E-4/,C12/-0.14452093E-7/
            DATA C13/6.5459673/
            T2=T*T
            T3=T2*T
            T4=T3*T
            IF(T.GT.273.15)GO TO 100
            PWSLN = C1/T+C2+C3*T+C4*T2+C5*T3+C6*T4+C7*ALOG(T)
            GO TO 200
        100 PWSLN = C8/T+C9+C10*T+C11*T2+C12*T3+C13*ALOG(T)
        200 PWSAT=EXP(PWSLN)
            RETURN
            END
      C=======================================================================
            FUNCTION TDBSAT(HSAT,PA)
      C=======================================================================
      C This function uses a table look-up using data from ASHRAE 1981
      C Fundamentals handbook, Page 6.5, table 2.
      C function is valid in the range of TDB from 32 to 100 F.
      C
      C TDBSAT = dry bulb temperature in degrees F at saturated conditions
      C PA  = atmospheric pressure in inches of mercury assummed to be 29.921 inhg
      C HSAT = enthalpy of saturated mixture in BTU per pound of dry air
      C
            REAL H(69),T(69)
            DATA H / 11.760,12.170,12.587,13.010,13.441,13.878,14.322,14.773,
           &         15.233,15.700,16.175,16.660,17.152,17.653,18.164,18.685,
           &         19.215,19.756,20.306,20.868,21.441,22.025,22.621,23.229,
           &         23.850,24.484,25.131,25.792,26.467,27.157,27.862,28.582,
           &         29.318,30.071,30.840,31.626,32.431,33.254,34.097,34.959,
           &         35.841,36.743,37.668,38.615,39.583,40.576,41.592,42.633,
           &         43.701,44.794,45.913,47.062,48.238,49.445,50.681,51.949,
           &         53.250,54.582,55.951,57.355,58.794,60.271,61.787,63.343,
           &         64.940,66.578,68.260,69.988,71.761/
            DATA T /   32.0,   33.0,   34.0,   35.0,   36.0,   37.0,   38.0,   39.0,
```

```
     &          40.0,  41.0,  42.0,  43.0,  44.0,  45.0,  46.0,  47.0,
     &          48.0,  49.0,  50.0,  51.0,  52.0,  53.0,  54.0,  55.0,
     &          56.0,  57.0,  58.0,  59.0,  60.0,  61.0,  62.0,  63.0,
     &          64.0,  65.0,  66.0,  67.0,  68.0,  69.0,  70.0,  71.0,
     &          72.0,  73.0,  74.0,  75.0,  76.0,  77.0,  78.0,  79.0,
     &          80.0,  81.0,  82.0,  83.0,  84.0,  85.0,  86.0,  87.0,
     &          88.0,  89.0,  90.0,  91.0,  92.0,  93.0,  94.0,  95.0,
     &          96.0,  97.0,  98.0,  99.0, 100.0/
C
      TDBSAT = TERPOL(T,H,HSAT,69)
      RETURN
      END
C======================================================================
      FUNCTION TERPOL(Y,X,XI,LTAB)
C======================================================================
C This is a general function to interpolate a value from a table.
C XI is the independent value, X is a list of independent values
C Y is a list of dependent values. LTAB is the length of the table.
C
      REAL Y(*),X(*)
C
C---------------------------------------------------------------------
C Locate position in table
C
      DO 1000 I = 1,LTAB
      IF((XI.GT.X(I).AND.XI.LT.X(I+1)).OR.
     &   (XI.LT.X(I).AND.XI.GT.X(I+1))) GO TO 2000
 1000 CONTINUE
C
C----------------------------------------------------------------------
C independant value supplied is not in table
C
      IF(X(LTAB).GT.X(1)) THEN
        IF(XI.LT.X(1)) THEN
          TERPOL = Y(1)
        ELSE IF(XI.GT.X(LTAB)) THEN
          TERPOL = Y(LTAB)
        ENDIF
      ELSE IF(X(1).GT.X(LTAB)) THEN
        IF(XI.GT.X(1)) THEN
          TERPOL = Y(1)
        ELSE IF(XI.LT.X(LTAB)) THEN
          TERPOL = Y(LTAB)
        ENDIF
      ELSE
        TERPOL = 0.0
      ENDIF
      RETURN
C
C----------------------------------------------------------------------
C interpolate
C
 2000 TERPOL = (XI-X(I)) / (X(I+1)-X(I)) * (Y(I+1)-Y(I)) + Y(I)
      RETURN
      END
```

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 85-3291	2. Performing Organ. Report No.	3. Publication Date JANUARY 1986

4. TITLE AND SUBTITLE

BUILDING EMULATION COMPUTER PROGRAM FOR TESTING OF ENERGY MANAGEMENT AND CONTROL SYSTEM ALGORITHMS

5. AUTHOR(S)

William B. May, Jr. and Cheol Park

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Naval Civil Engineering Laboratory
U.S. Department of Defense
Port Hueneme, California 93043

Office of Building and Community Systems
U.S. Department of Energy
1000 Independence Avenue, SW
Washington, DC 20585

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

A building emulator can be used to test energy management and control systems (EMCS). The emulator uses a computer program to simulate the responses of a building including the equipment, building space, and building envelope to EMCS commands. Building model software for the emulator has been developed at the National Bureau of Standards (NBS) in an effort to assist the United States Naval Civil Engineering Laboratory (NCEL), which is developing a sophisticated building emulator.

The concept of the building emulator and the building emulator computer program are described in this report. The program includes the weather, the air handling unit, the zone, and the comfort model. In addition, the energy compilation routine is also included. The models presented here are simplified models. With these abridged models, a single zone building with exterior walls and a single deck air handling unit are simulated. A complete FORTRAN source code of the building emulator computer program is appended.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

air handling unit simulation model; building emulator; building space zone model; computer simulation; EMCS algorithms; energy management and control systems (EMCS); HVAC emulator; local equipment simulation model; testing EMCS algorithms.

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

132

15. Price

$16.95