-3285

Verification of Public Domain Control Algorithms for Building Energy Management and Control Systems

William B. May, Jr.
George E. Kelly

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

December 1985

nsored by:
. Naval Civil Engineering Laboratory
. Department of Energy

NBSIR 85-3285

VERIFICATION OF PUBLIC DOMAIN CONTROL ALGORITHMS FOR BUILDING ENERGY MANAGEMENT AND CONTROL SYSTEMS

William B. May, Jr.
George E. Kelly

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:
U.S. Naval Civil Engineering Laboratory
U.S. Department of Energy

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

ABSTRACT

Software is an important component of building energy management and control systems (EMCS). It is usually supplied by EMCS vendors in proprietary packages that do not contain human readable source code. Even when source code listings are made available on a "limited use" basis, it is difficult for the user to determine whether the supplied algorithms meet the design specifications because of the lack of public domain HVAC control algorithms with which to compare them.

To help overcome the above problem, the National Bureau of Standards developed and documented eight public domain EMCS supervisory control algorithms. The testing and verification of these eight algorithms are described in this report. The algorithms tested cover dry bulb and enthalpy economizer cycles, optimum and scheduled start/stop, duty cycling, demand limiting, outside air supply air reset, and demand supply air reset. For each of these algorithms, the process of installing the algorithms on an NBS laboratory system is discussed and a description is given of the tests performed. The results of these experimental studies are presented, along with any additional considerations for use of the algorithms that were developed as a result of the testing program.

Key words: algorithm verification; control algorithms; demand limiting; duty cycling; economizer cycles; energy management algorithms; field testing; HVAC control; optimum start/stop; temperature reset

ACKNOWLEDGEMENTS

The authors of this report would like to acknowledge the assistance of a number of people and organizations without whose cooperation the results presented in the report would not have been possible. The work was funded by the National Bureau of Standards, the U.S. Naval Civil Engineering Laboratory, Department of Defense, and the U.S. Department of Energy. Dr. Cheol Park developed many of the public domain algorithms tested in this report. The NBS Plant Air Conditioning Shop patiently allowed the testing of algorithms on an air handling unit for which they were responsible and provided assistance with instrumentation. Warren Hurley provided assistance with instrumentation and acted as liaison with the Air Conditioning Shop. J.M. Callen provided invaluable assistance in conducting tests of the duty cycling algorithm. Special acknowledgement is also due to the occupants of the offices in the quadrant of building 226 at NBS which was supplied by the air handling unit used for testing the algorithms, particularly Jan Clark and Laurie Watkins.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CONVERSION FACTORS FROM ENGLISH TO METRIC (SI) UNITS

Physical Characteristic	To Convert from	To	Multiply by
Length	ft	m	0.3048
Area	ft^2	m^2	0.0929
Velocity	m/s	mph	0.447
Temperature	^oF	^oC	$t_c = (t_F - 32)/1.8$
Temperature difference	^oF	^oC	0.55555
Energy	Btu	J	1.055×10^3
Power	Btu/hr	W	0.293
U-value	$Btu/hr \cdot ft^{2} \cdot ^oF$	$W/m^{2}{}^oC$	5.678
Thermal resistance	$hr \cdot ft^{2} \cdot ^oF/Btu$	$m^{2}{}^oC/W$	0.1761
Pressure	in Hg 60^oF	KPa	3.376

1. INTRODUCTION

A computer-based building energy management and control system (EMCS) relies heavily on computer software to utilize efficiently the heating, ventilating, and air conditioning (HVAC) equipment in the building. HVAC control software is available in proprietary or system dependent packages, usually supplied with a EMCS system supplied by a particular vendor. However, if listings of the software source are not supplied, the EMCS owner or HVAC designer will not know if the EMCS system can meet the HVAC design specifications. Even if the control algorithms used are known, there is not much HVAC control algorithm software in the public domain to use for baseline comparison purposes. This report describes the field testing of non-proprietary algorithms developed at the National Bureau of Standards (NBS) in the NBS building management and controls laboratory.

EMCS control strategies fall into at least two categories. Direct control (or Direct Digital Control, DDC) refers to strategies and algorithms that control the building equipment directly without the use of conventional pneumatic or electronic local analog control, implementing lower level functions such as closed loop control of valves, dampers, and actuators. Supervisory or management strategies control the building in a broader sense, managing equipment systems by methods such as selecting setpoints, and choosing optimum operating times as a function of variables including electrical demand, time, weather conditions, occupancy, and heating and cooling requirements.

This report describes the testing of several types of supervisory control strategies. These are listed in table 1-1. The algorithms were all developed at NBS and are therefore in the public domain. The description of the development and operation of the algorithms has been fully described in a series of reports [1,2,3,4,5].

The testing described in this report was intended only to verify that the algorithms developed would function as expected when installed on an actual EMCS system controlling HVAC equipment in an actual building. No attempt has been made to quantify the amount of energy saved by using the algorithms since the instrumentation on the HVAC equipment was not sufficient to measure enough variables to calculate energy values. In general the tests were carried out on a single building air handling unit and any energy conservation results would be of limited general value for evaluating the usefulness of these algorithms for energy conservation.

The EMCS algorithms were all tested using an actual EMCS system developed in the laboratory at NBS. This EMCS is described in section 1.1 below. The EMCS was connected to and controlled an air handling unit in an actual building. The air handling unit is described in section 1.2 below. The first step in the testing procedure for the algorithms was to install the algorithm in the NBS EMCS with the goal of controlling some aspect of the air handling unit operation. It should be kept in mind that once the algorithm was installed on

the specific EMCS system, it was no longer considered to be the actual general algorithm but was then considered to be an implementation of the generalized algorithm. If problems were encountered while installing or initially testing the algorithm implementation, an attempt was made to modify the implementation to correct the problem. Problems which appeared to be of a fundamental nature are described in the report.

table 1-1. public domain algorithms tested

algorithm name	purpose	software level
1. Dry Bulb Economizer	reduce cooling load with outside air using value of outdoor dry bulb temperature.	field processor
2. Enthalpy Economizer	reduce cooling load with outside air using comparison of outside and return air enthalpies.	field processor
3. Optimum Start/stop	start and stop HVAC equipment so that comfort conditions are only maintained during the occupied period.	central EMCS computer
4. Scheduled Start/stop	start and stop HVAC equipment at scheduled times of day on selected days of the week.	field processor
5. Duty Cycling	reduce energy use by turning off equipment for short periods of time during times of lower loads.	field processor
6. Demand Limiting	minimize electrical demand charges by turning off electrical loads if certain electrical demand limits are exceeded.	central EMCS computer
7. Outside Supply Air Reset	adjust setpoint of supply air temperature from an air handling unit to match loading by using outside air temperature as an indirect indicator of load.	field processor
8. Demand Supply Air Reset	adjust setpoint of supply air temperature from an air handling unit to match loading by using measurements of space load.	field processor

Chapters 2 through 9 of this report are each concerned with one of the algorithms in table 1-1. Each chapter begins with a brief description of the operation of the algorithm and this is followed by a discussion of what steps were involved in installing the algorithm on the EMCS system and using the algorithm to control the air handling unit. Any changes that were made to the original algorithms as described in the original reports are noted in a separate section in each chapter. Following the description of changes, each chapter contains a description of the nature of verification testing used for each algorithm. Results of verification testing are included in each chapter in the form of tables, plots, and descriptions. Another section in the chapter for each algorithm contains considerations for use of the algorithm which were developed in the course of testing. The considerations take the form of either suggestions and rules for determining values for algorithm parameters or general observations about the usage of the algorithm.

1.1 EMCS System Used for Verification

The Building EMCS used for verification testing of the public domain algorithms was not a commercial or proprietary system. It was developed in-house by staff of the NBS building systems and controls program. The system utilized distributed computer processors arranged in a hierarchy as shown in figure 1-1.

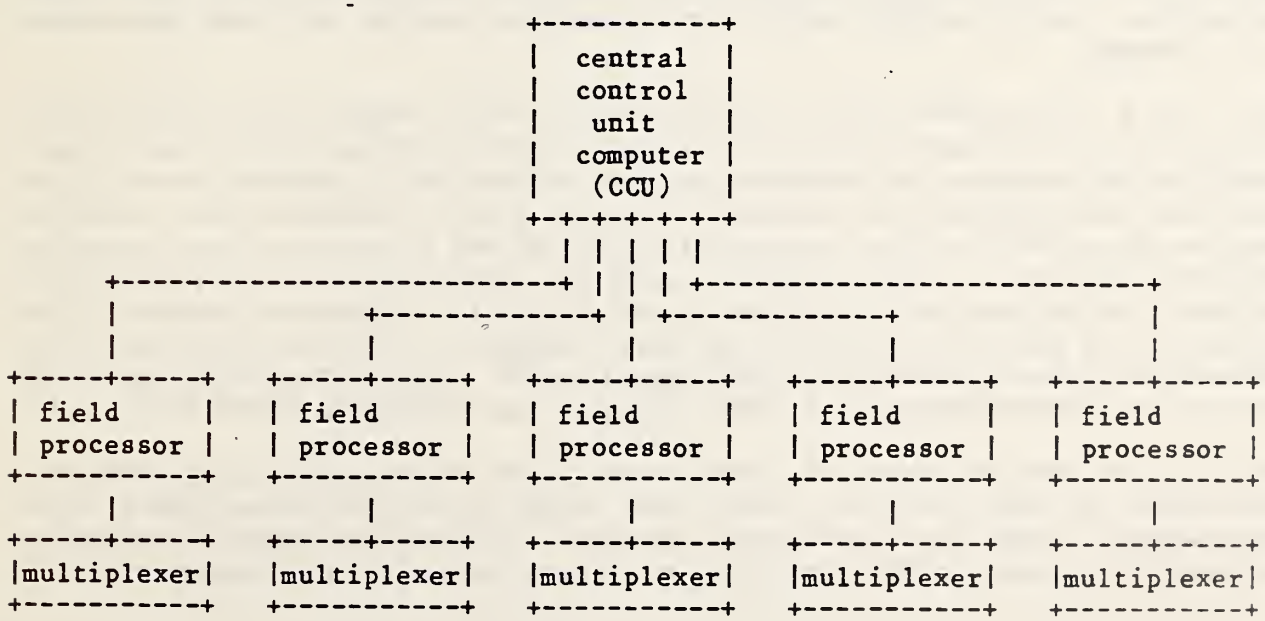

figure 1-1. schematic diagram of NBS laboratory EMCS processors

The multiplexers shown in figure 1-1 are microcomputers used to receive signals from the sensors of the EMCS and convert them into digitally coded numbers. The sensors include thermistors, pressure transducers, dew point sensors, and switch contacts. The multiplexers are connected to the next level in the hierarchy, the field processor, by a serial communications line. The multiplexers can send reports on the state of the EMCS sensors up to the field processors upon receiving a command to do so from the field processors. The field processors can also send commands to control digital and analog outputs. Upon receiving such commands the multiplexer will control digital switches or relays or set signals on analog outputs. The output devices connected to the multiplexer include motorized bi-directional pressure regulators for pneumatic control of valve and damper actuators, and electrical relays.

The field processors shown in figure 1-1 are also microcomputers. These devices contain software to communicate with the multiplexers and make control decisions based on information from the sensors connected to the multiplexers. The field processor software contains implementations of EMCS control algorithms. The most important algorithm in the field processor is used to provide DDC of the valves and dampers on an air handling unit to maintain the supply air temperature at a setpoint.

The central control unit computer (CCU) shown in figure 1-1 is connected by serial communication lines to all of the field processors. The CCU computer is used to allow collection and storage of historical data from the EMCS sensors, detect and display alarm conditions, control configuration of the field processors, and allow EMCS algorithm parameters used by the field processors to be changed.

The field processors and multiplexers were constructed at NBS from commerically available computer circuit boards inserted into a bus. Signal conditioning hardware was constructed from commercial integrated circuits and discrete components. The software in the field processor was written completely at NBS in the language FORTRAN IV and microprocessor assembly language. Changes to algorithms were made by altering the source program and reconstructing the field processor software on a separate computer. The revised field processor software was then loaded into the field processor. The CCU computer was a commercial minicomputer with a proprietary disk operating system. The EMCS software on the CCU computer was written in FORTRAN 77.

The last column in table 1-1 shows where each of the EMCS algorithms was installed in order to run tests. The majority of the algorithms were implemented in the field processor software. Software for changing algorithm parameters for each of the algorithms was installed on the CCU computer.

1.2 Air Handling Unit Used for Verification

The air handling unit used in the algorithm testing is one of seven air handlers located in the attic mechanical room of one of the office/laboratory buildings on the campus of NBS in Gaithersburg, MD. The unit provides approximately 9.4 m3/s of primary air to 39 office modules (total of 618 m2 floor area) located on the building perimeter. Three fluid-to-air heat exchange coils are contained within the air handler, two of the coils providing pre-heat with steam, and the remaining coil cooling the air stream with chilled water. Air entering the unit is a mixture of return and outside air controlled by dampers. The flow of steam or water through the coils was controlled by pneumatically actuated valves connected to the EMCS. The outside air dampers were also pneumatically actuated and under control by the EMCS.

The offices in the building supplied by the air handling unit contained induction terminal reheat type local conditioning equipment with a hot water heating coil under the control of a local thermostat. Conditioned air from the air handling unit was supplied through the terminal reheat unit. Return air to the air handling unit was taken from return ducts located in the ceiling of the halls in the building. Each office had grills over the door to allow air flow from the room to the return duct.

2. VERIFICATION OF A DRY BULB ECONOMIZER ALGORITHM

A dry bulb economizer algorithm is used to reduce the need for mechanical cooling in a building when the outside air is in general cooler than the air in the building spaces. This algorithm is usually applied to the control of an air handling unit supplying air at a setpoint temperature to building zones. The air handling unit will have dampers which can be controlled to cause air entering the air handling unit to come from the air returning from the building interior or from the outside or from a mixture of the two. The dry bulb economizer algorithm uses a measured value of the outside air dry bulb temperature to determine when outside air should and should not be used for cooling. A detailed description of the public domain dry bulb economizer algorithm developed at NBS may be found in a previous report [4].

2.1 Operation of Algorithm

Operation of the dry bulb economizer algorithm is described completely in reference [4]. However, a brief description of the algorithm operation will be given here. The dry bulb algorithm uses one measured value from the EMCS, the outside dry bulb temperature. In addition the algorithm is assumed to be able to access the current supply air setpoint temperature for the air handling unit. Two algorithm parameters must be supplied and these are the changeover temperature and the minimum temperature. The algorithm has four possible output decisions. These are:

1. lock outside air dampers closed and use mechanical cooling.
2. lock outside air dampers closed and do not use mechanical cooling.
3. lock outside air dampers open and also use mechanical cooling.
4. allow outside air dampers to be modulated and do not use mechanical cooling.

If the outside temperature is greater than the changeover temperature then decision 1 above is made. If the outside temperature is less than the minimum temperature then decision 2 is made. If the outside temperature is less than the changeover temperature, greater than the minimum temperature and greater than the supply air setpoint then decision 3 is made. If the outside temperature is less than the changeover temperature, greater than the minimum temperature, and less than the setpoint then decision 4 is made.

2.2 Algorithm Installation

The public domain dry bulb economizer algorithm was installed on the NBS laboratory EMCS. The algorithm was installed in a field processor by integrating it into the field processor software. The algorithm reported in reference [4] was originally implemented in FORTRAN 77. In order to install the algorithm in the field processor software, the algorithm had to be rewritten in FORTRAN IV, the high level language used to implement algorithms

in the field processor. Other than this change, the algorithm was not modified. The algorithm was installed in a FORTRAN IV subroutine which was called by an air handling unit control task. The air handling unit controller performed direct digital control of the valves and dampers in the air handling unit and provided automatic sequencing between control of the valves and dampers under changing load conditions. The four actions described in section 2.1 were implemented as:

1. lock outside air dampers closed and use mechanical cooling: outside air dampers were forced closed and were not allowed to be used for cooling. The cooling coil was selected for DDC.
2. lock outside air dampers closed and do not use mechanical cooling: outside air dampers were forced closed and were not allow to be used for cooling.
3. lock outside air dampers open and also use mechanical cooling: outside air dampers were forced open and the cooling coil was selected for DDC.
4. allow outside air dampers to be modulated and do not use mechanical cooling: the cooling coil valve was forced closed and the outside air dampers were selected for DDC.

2.3 Corrections to Original Algorithm

No problems were discovered with the original dry bulb economizer algorithm as written. Therefore no corrections were made to the algorithm.

2.4 Testing of Algorithm

The testing of the public domain dry bulb economizer algorithm consisted of allowing the algorithm to operate in the NBS laboratory EMCS field processor which was controlling an actual air handling unit with direct digital control. The air handling unit was supplying air to actual offices during the testing. The unit was instrumented with temperature sensors in the supply, return, outdoor, and mixed air, dew point sensors in the outdoor and mixed air, and relative humidity sensors in the supply and return air. In addition, the cooling coil valve and outside air dampers were instrumented with position sensors. Data were taken from these sensors and others at five minute intervals and recorded on the EMCS central computer.

2.4.1 Test Description

For evaluation of the dry bulb economizer algorithm data were collected continously from the NBS laboratory EMCS between April 7 and April 30, 1985 at five minute intervals, resulting in over 6700 data readings. During this period outside air temperatures varied between 25 F and 90 F. The changeover

temperature for the algorithm was set at 68 F, and the minimum temperature was set at 42 F.

2.4.2 Economizer Algorithm Performance

The public domain dry bulb economizer algorithm was actually installed in the NBS laboratory EMCS in April 1984, and had been operating continously in the control of the test air handling unit connected to actual building spaces since that time without any difficulties.

Figure 2-1 presents results of testing the dry bulb economizer during the period between April 7 - 30, 1985. During that time, the state of the outside air dampers was recorded every five minutes. In preparing figure 2-1, the outside air dampers were assumed to be in one of four possible states. These states were completely open, completely closed, between half open and fully open, and between half open and fully closed. Figure 2-1 contains four histograms to indicate the number of times that the outside air dampers were in each of the four states. The number of occurrences of each outside air damper state is plotted as a function of the outside dry bulb temperature. The figure shows that when the outside air temperature was between 60 and 66 F the outside air dampers were always open.

Theoretically, if the algorithm were operating properly, the outside air damper should always be closed above the changeover temperature and open just below the changeover temperature. In figure 2-1, a slight overlap of the two histograms is observed. There were 5 occurences where the temperature was between 67.0 F and 67.5 F and the damper was closed, and 46 occurrences where the temperature was between 67.5 F and 68.0 F where the damper was closed. There were 8 occurrences where the temperature was between 68.5 F and 69.0 F and the dampers were open, and 23 occurrences where the temperature was between 68.0 and 68.5 F and the dampers were open. These errors total 82 occurrences. This represents approximately one percent of the testing period. Unfortunately, the data shown do not represent only the performance of the dry bulb algorithm, but also include the effects of errors in the EMCS control of the air handling unit.

The performance of the algorithm in keeping the dampers closed when the outside air temperature is below the minimum temperature can also be observed in figure 2-1. There were no occurrences where the outside air dampers were open at all when the dry bulb temperature was below the 42 F minimum temperature.

Figure 2-1 illustrated that the public domain dry bulb economizer algorithm performed as expected. In order to evaluate the performance of the algorithm in minimizing energy consumption, figure 2-2 was prepared. Figure 2-2 has histograms for the four states of the outside air damper as did figure 2-1, but the frequency in figure 2-2 is plotted as a function of the measured difference between the outside air and return air enthalpies (enthalpies were

calculated from dry bulb temperature, barometric pressure, and either dew point or relative humidity). Theoretically, if a perfect enthalpy economizer algorithm were being employed instead of this particular dry bulb economizer algorithm, the outside air damper should always be closed when the outside air enthalpy was greater than the return air enthalpy and open when the outside air enthalpy was just below the return air enthalpy temperature. In figure 2-2 it may be observed that the dampers were not open when the outside air enthalpy was greater than the return air enthalpy, but the dampers were often closed when the outside air enthalpy was below the return air enthalpy. This area of the damper closed histogram below the zero enthalpy difference point represents a wasted potential for energy savings of this type, or approximately eight percent of the test period. This waste is due to three effects. The first effect is from possible errors in the EMCS control of the air handling unit external to the dry bulb economizer algorithm. The second effect is from a possible poor choice of the changeover temperature, and the third effect is due to the basic assumptions made by the dry bulb economizer that the outside air remains at a constant moisture content.

Figure 2-3 is a plot of the difference in enthalpy between the outside and return air as a function of the outside air temperature for all data taken during the test period. If the moisture content of the return air and the outside air were constant the relationship between the two variables would be a straight line. The figure shows that the relationship is reasonably linear but that there is a spread in the enthalpy differential for any dry bulb temperature of approximately two Btu per pound of air. This spread indicates that it is not possible to obtain perfection with the dry bulb economizer algorithm and that the performance of the algorithm relative to the ideal depends on the statistics of the weather.

Figure 2-3 also shows that for the test period, a zero enthalpy differential approximately corresponds to an outside dry bulb temperature of 72 F. This indicates that for this test period, a changeover temperature of 72 F would probably have resulted in better dry bulb algorithm performance.

2.5 Considerations in Use of Algorithm

2.5.1 Parameter Selection

The proper changeover temperature for the dry bulb economizer algorithm depends on the weather conditions for a building and the building loads (which determine the return air temperature variations). If data which allow calculation of outside and return air enthalpy are available a plot such as shown in figure 2-3 can aid in selection of the changeover temperature. The ideal changeover temperature will change with time and can either be left at a constant value or adjusted with time. Some sophisticated EMCS systems may be able to adjust this parameter automatically.

The minimum air temperature is a compromise between the occurrence of equipment damage due to freezing and maximum energy conservation. In the testing, the air handling unit loads did not require a significant amount of cooling at temperatures below the selected minimum. In buildings with large internal loads, the lowest minimum temperature which will prevent freezing would be desirable to minimize energy consumption.

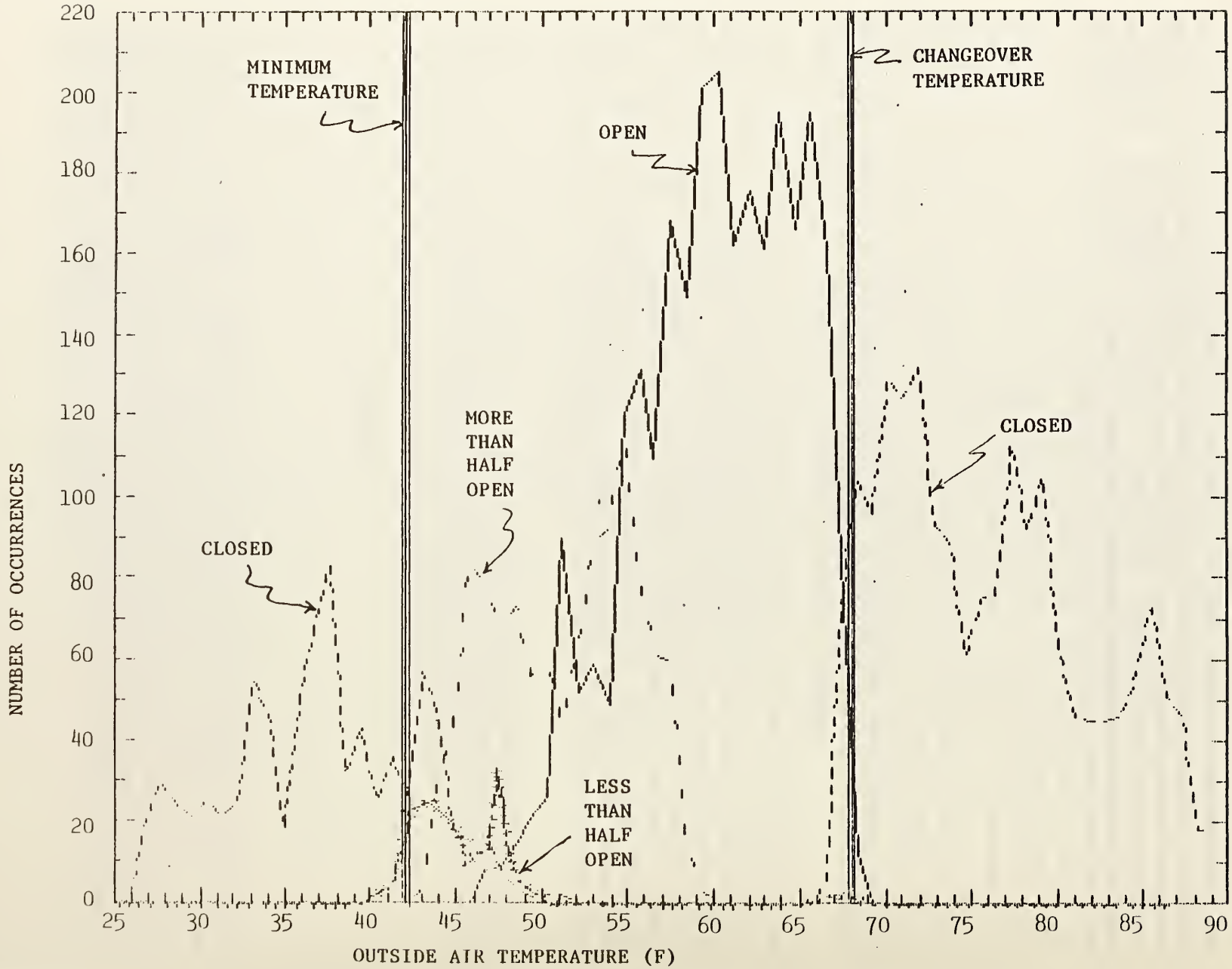

Figure 2-1. Results of testing the dry bulb economizer algorithm between April 7 and April 30, 1985

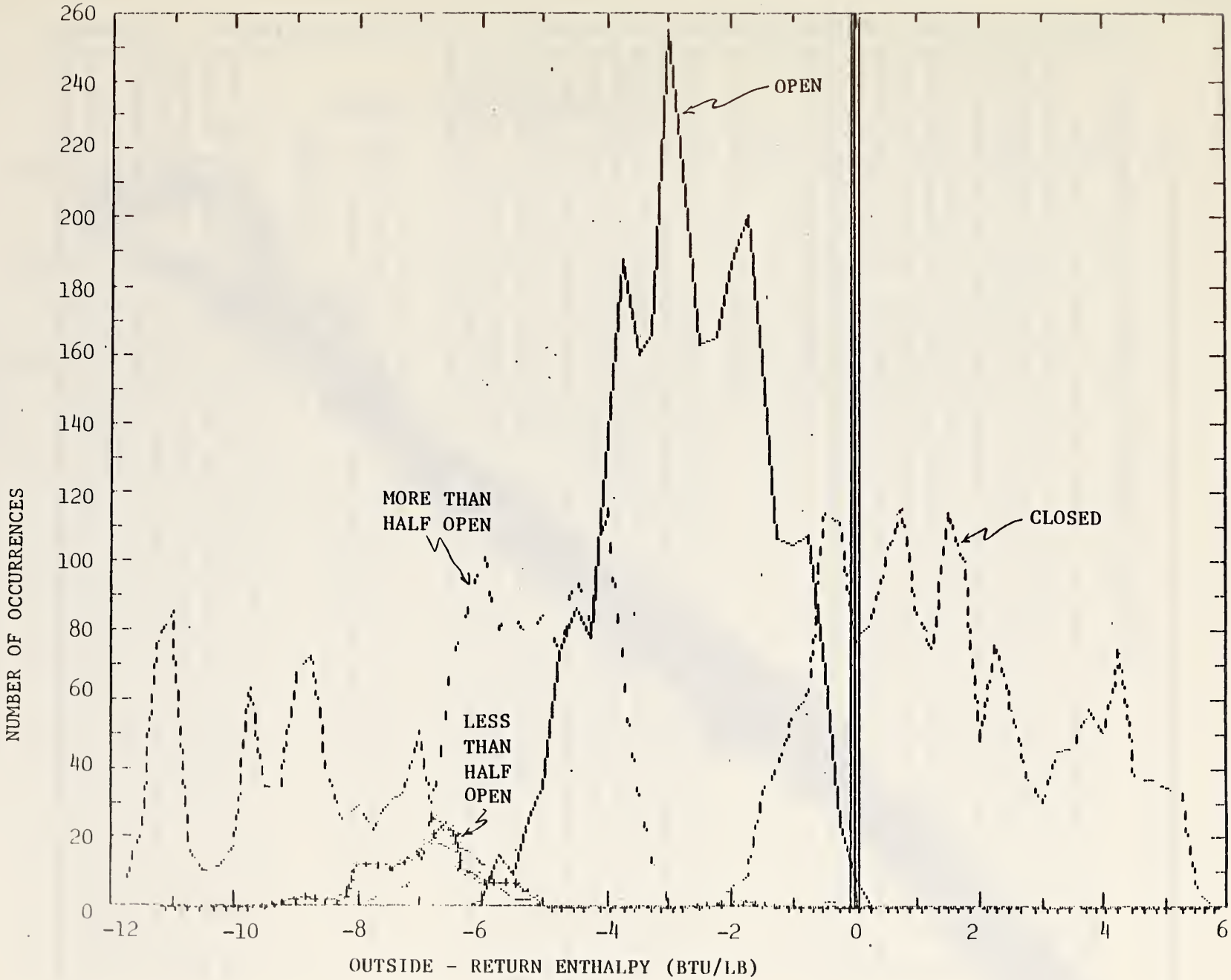

Figure 2-2. Performance of dry bulb economizer algorithm

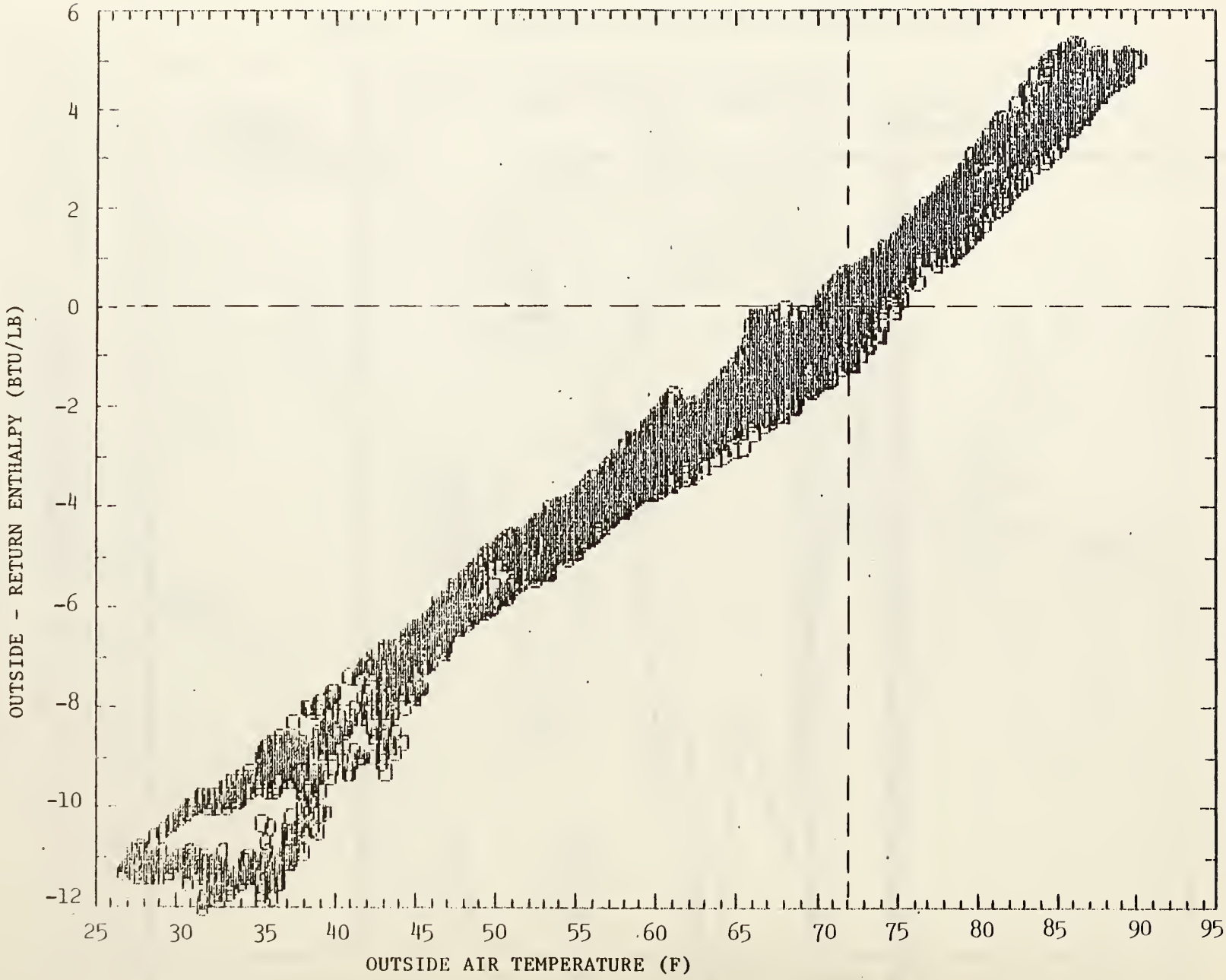

Figure 2-3. Enthalpy difference as a function of outside temperature (April 7-30, 1985)

3. VERIFICATION OF AN ENTHALPY ECONOMIZER ALGORITHM

An economizer algorithm is used to reduce the need for mechanical cooling in a building when the outside air is in general cooler than the air in the building spaces. This algorithm is usually applied to the control of an air handling unit supplying air at a setpoint temperature to building zones. The air handling unit will have dampers which can be controlled to cause air heated or cooled in the handling unit to come from the air returning from the building interior or from the outside or from a mixture of the two. An enthalpy economizer algorithm compares the enthalpy of the outside air with the enthalpy of the air returning from the building spaces to determine when outside air should and should not be used for cooling. A detailed description of the public domain enthalpy economizer algorithm developed at NBS may be found in a previous report [4].

3.1 Operation of the Algorithm

Operation of the enthalpy economizer algorithm is described completely in reference [4]. However, a brief description of the algorithm operation will be given here. The enthalpy algorithm uses four measured values from the EMCS in order to be able to determine enthalpies of the outside air and the return air temperature from the building spaces. The four measured values are for the outside dry bulb temperature, the return air dry bulb temperature, the outside relative humidity or dew point temperature and the return air relative humidity or dew point temperature. In addition, the algorithm is assumed to be able to access the current supply air setpoint temperature for the air handling unit. One algorithm parameter, the minimum temperature, must be supplied. The algorithm has four possible output decisions. These are:

1. lock outside air dampers closed and use mechanical cooling.
2. lock outside air dampers closed and do not use mechanical cooling.
3. lock outside air dampers open and also use mechanical cooling.
4. allow outside air dampers to be modulated and do not use mechanical cooling.

If the outside air enthalpy is greater than the return air enthalpy by at least a small differential (set at 0.5 Btu/lb of air during the testing) then decision 1 above is made. If the outside temperature is less than the minimum temperature then decision 2 is made. If the outside air enthalpy is less than the return air enthalpy by at least a small differential, greater than the minimum temperature and greater than the supply air setpoint then decision 3 is made. If the outside air enthalpy is less than the return air enthalpy by at least a small differential temperature, greater than the minimum temperature, and less than the setpoint then decision 4 is made.

3.2 Installation of Algorithm

The public domain enthalpy economizer algorithm was installed on the NBS laboratory EMCS. The algorithm was installed in a field processor, integrated into the field processor software. The algorithm reported in reference [4] was originally implemented in FORTRAN 77. In order to install the algorithm in the field processor software, the algorithm had to be rewritten in FORTRAN IV, the high level language used to implement algorithms in the field processor. The subroutine used by the algorithm for psychrometric calculations was shortened by removing a section which allowed calculations to be performed in either English or SI units, leaving only the English units calculations. Other than these changes, the algorithm as originally published [4] was not modified. The algorithm was installed in a FORTRAN IV subroutine which was called by an air handling unit control task. The air handling unit controller performed direct digital control of the valves and dampers in the air handling unit and provided automatic sequencing between control of the valves and dampers under changing load conditions. The four actions described in section 3.1 were implemented as:

1. lock outside air dampers closed and use mechanical cooling: outside air dampers were forced closed and were not allowed to be used for cooling. The cooling coil was selected for DDC.
2. lock outside air dampers closed and do not use mechanical cooling: outside air dampers were forced closed and were not allow to be used for cooling.
3. lock outside air dampers open and also use mechanical cooling: outside air dampers were forced open and the cooling coil was selected for DDC.
4. allow outside air dampers to be modulated and do not use mechanical cooling: the cooling coil valve was forced closed and the outside air dampers were selected for DDC.

3.3 Corrections to Original Algorithm

No problems were discovered with the original enthalpy economizer algorithm as written. Therefore no corrections were made to the algorithm.

3.4 Testing of Algorithm

The testing of the public domain enthalpy economizer algorithm consisted of allowing the algorithm to operate in the NBS laboratory EMCS field processor which was controlling an actual air handling unit with direct digital control. The air handling unit was supplying air to actual offices during the testing. The unit was instrumented with temperature sensors in the supply, return, outdoor, and mixed air, dew point sensors in the outdoor and mixed air, and relative humidity sensors in the supply and return air. In addition, the cooling coil valve and outside air dampers were instrumented with position

sensors. Data were taken from these sensors and others at five minute intervals and recorded on the EMCS central computer.

3.4.1 Test Procedure

For evaluation of the enthalpy economizer algorithm, data were collected continously from the NBS laboratory EMCS between May 2 and May 18, 1985 at five minute intervals, resulting in over 3700 data readings. During this period outside air temperatures varied between 43 and 85 F. The minimum temperature parameter for the algorithm was set at 42 F.

3.4.2 Economizer Algorithm Performance

Figure 3-1 presents results of testing the enthalpy economizer during the period between May 2 - 18, 1985. During that time, the state of the outside air dampers was recorded every five minutes. In preparing figure 3-1, the outside air dampers were assumed to be in one of four possible states. These states were completely open, completely closed, between half open and fully open, and between half open and fully closed. Figure 3-1 contains four histograms to indicate the number of times that the outside air dampers were in each of the four states. The number of occurrences of each state is plotted as a function of the measured difference between the outside air and return air enthalpies (enthalpies were calculated from dry bulb temperature, barometric pressure, and either dew point or relative humidity). Theoretically, if the algorithm were operating properly, the outside air damper should always be closed when the outside air enthalpy is greater than the return air enthalpy and open when the outside air enthalpy is just below the return air enthalpy temperature. In figure 3-1 it may be observed that in some cases the dampers were open when the outside air enthalpy was greater than the return air enthalpy, and also that the dampers were sometimes closed when the outside air enthalpy was below the return air enthalpy. However, the algorithm allowed a differential of 0.5 Btu/lb of air around the zero difference in enthalpy point. With one exception, all of the situations where the damper is in the wrong position lie within this differential. There was one data point where the enthalpy difference was between 0.5 and 0.75 Btu/lb. These results indicate that the algorithm performed properly in selecting whether or not to use outside air for cooling.

In section 2, which described testing of a dry bulb economizer algorithm, the number of occurrences of each outside air damper state were plotted as a function of the outside dry bulb temperature. For consistency, the enthalpy economizer results are also plotted in this form in figure 3-2. The figure shows that there is no single dry bulb temperature where the outside air damper is always closed above this temperature and always open just below this temperature. Instead, for this particular test period, there was a spread of temperatures between 66 F and 74 F for which the outside air damper could be either in the open or closed position.

The performance of the algorithm in keeping the dampers closed when the outside air temperature is below the minimum temperature can be observed in figure 3-2. There were no occurrences where the outside air dampers were open at all when the dry bulb temperature was below the 42 F minimum temperature.

Figure 3-3 is a plot of the difference in enthalpy between the outside and return air as a function of the outside air temperature for all data taken during the test period. If the moisture content of the return air and the outside air were constant the relationship between the two variables would be a straight line. The figure shows that the relationship is reasonably linear but that there is a spread in the enthalpy differential for any dry bulb temperature of approximately two Btu per pound of air. Figure 3-3 can be compared to figure 2-3 which is the same type of graph for a different period of time. In both figures the spread in enthalpy difference is similar. Figure 2-3 showed that a zero enthalpy differential approximately corresponded to an outside dry bulb temperature of 72 F. In figure 3-3, the zero enthalpy difference approximately corresponds to 70 F.

3.5 Considerations for Use of Algorithm

3.5.1 Parameter Selection

There are no parameters to select for the public domain enthalphy economizer algorithm other than the minimum temperature which can be set one time and left unchanged. The dry bulb economizer algorithm has parameters which have an optimum value that depends on the weather conditions and which will change with time. This is the advantage of an enthalpy economizer algorithm over a dry bulb economizer algorithm.

3.5.2 Usage Constraints

The disadvantage of the enthalpy enconomizer compared to a dry bulb economizer is that additional instrumentation for the measurement of air moisture content is required. At least one additional temperature sensor and two moisture measurement sensors must be installed. During the operation of economizer algorithms on the NBS laboratory EMCS, a number of problems were encountered in maintaining the moisture sensors. These problems usually involved the measurement of high humidity air which caused failure in the sensors. Regular maintenance was required for dew point sensors to keep them clean. A decision must usually be made which involves the comparison of an incremental energy savings resulting from the use of an enthalpy economizer algorithm over a dry bulb economizer algorithm with the increased costs of installation and maintenance of moisture measurement sensors.

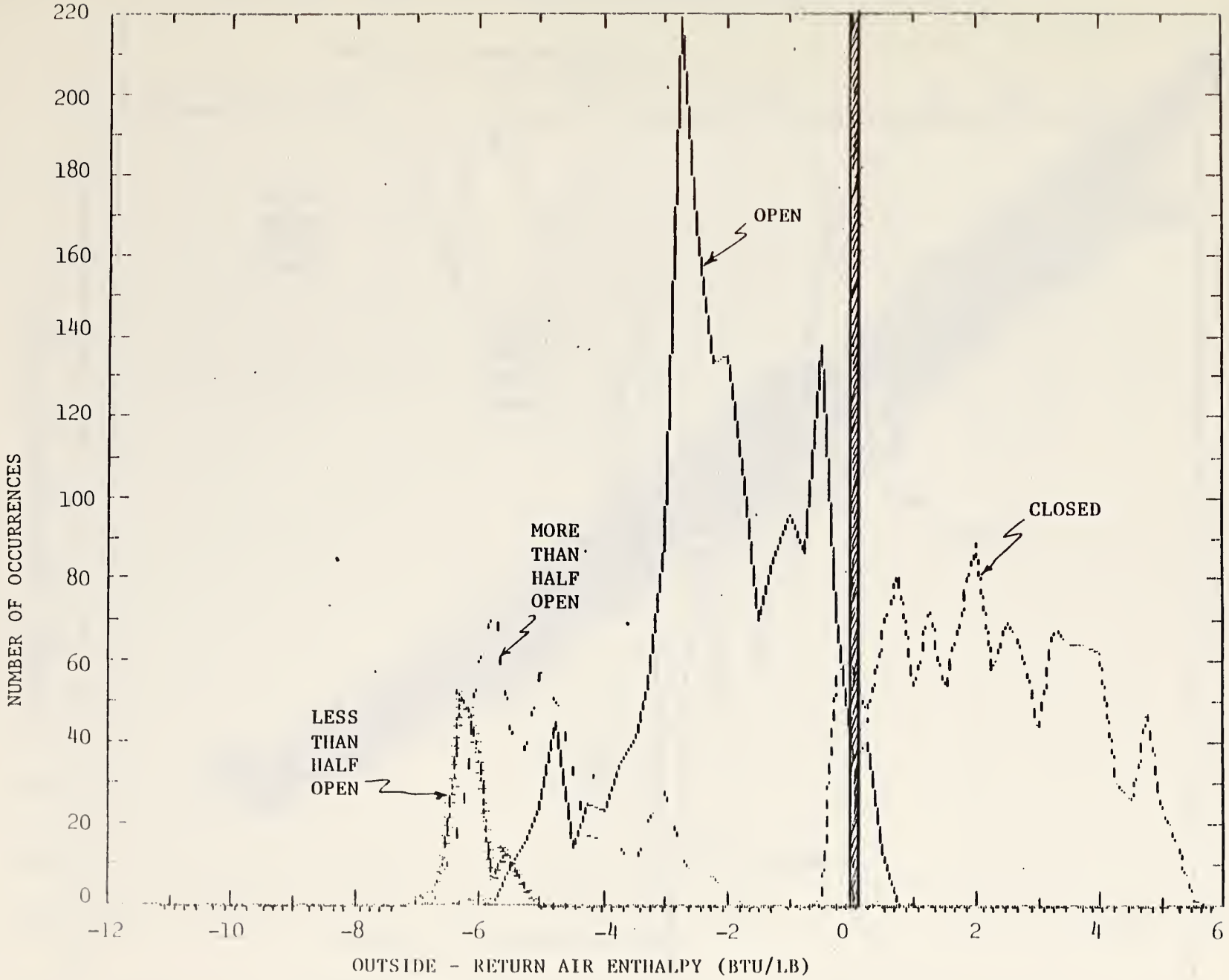

Figure 3-1. Performance of enthalpy economizer algorithm

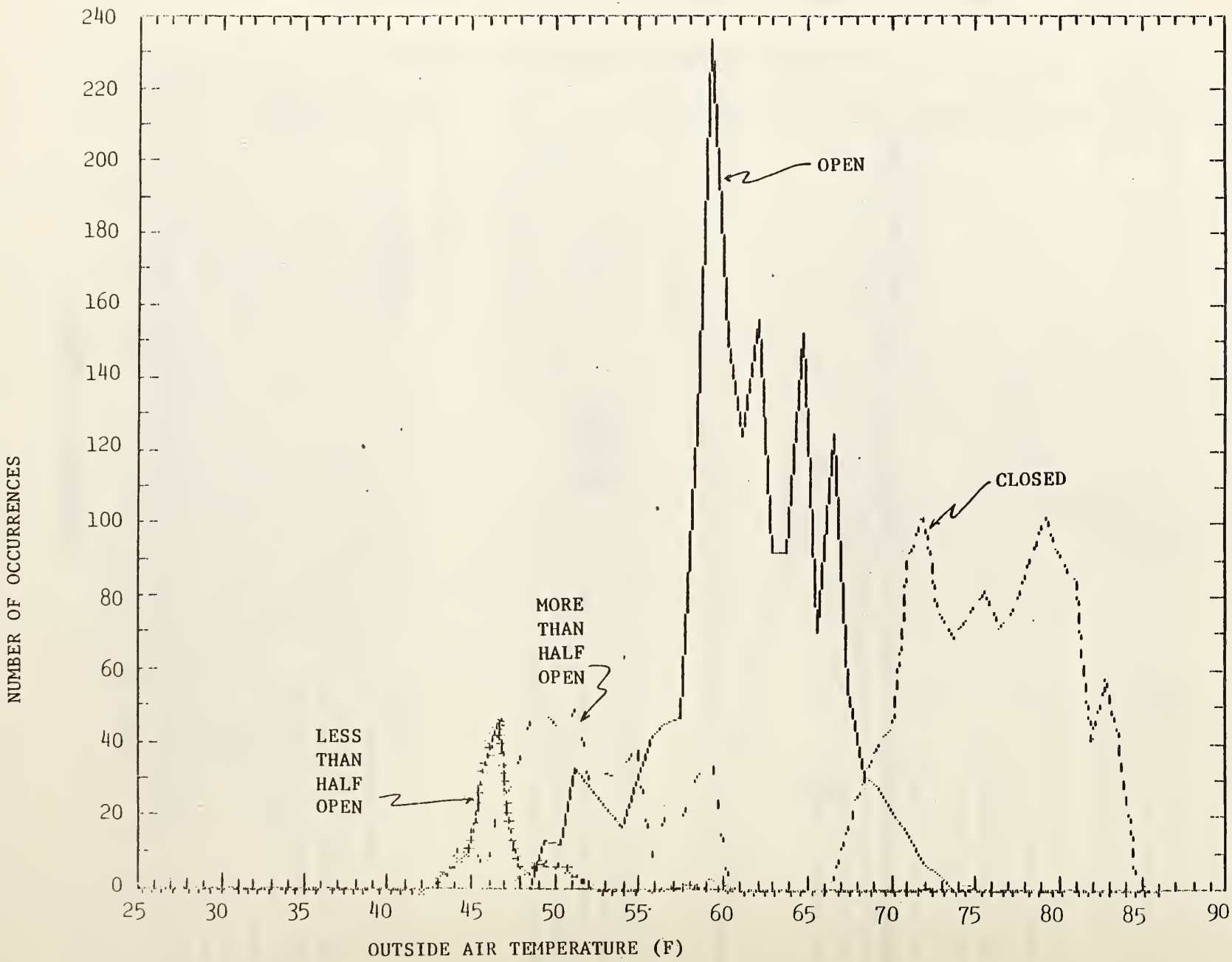

Figure 3-2. Results of testing the enthalpy economizer algorithm

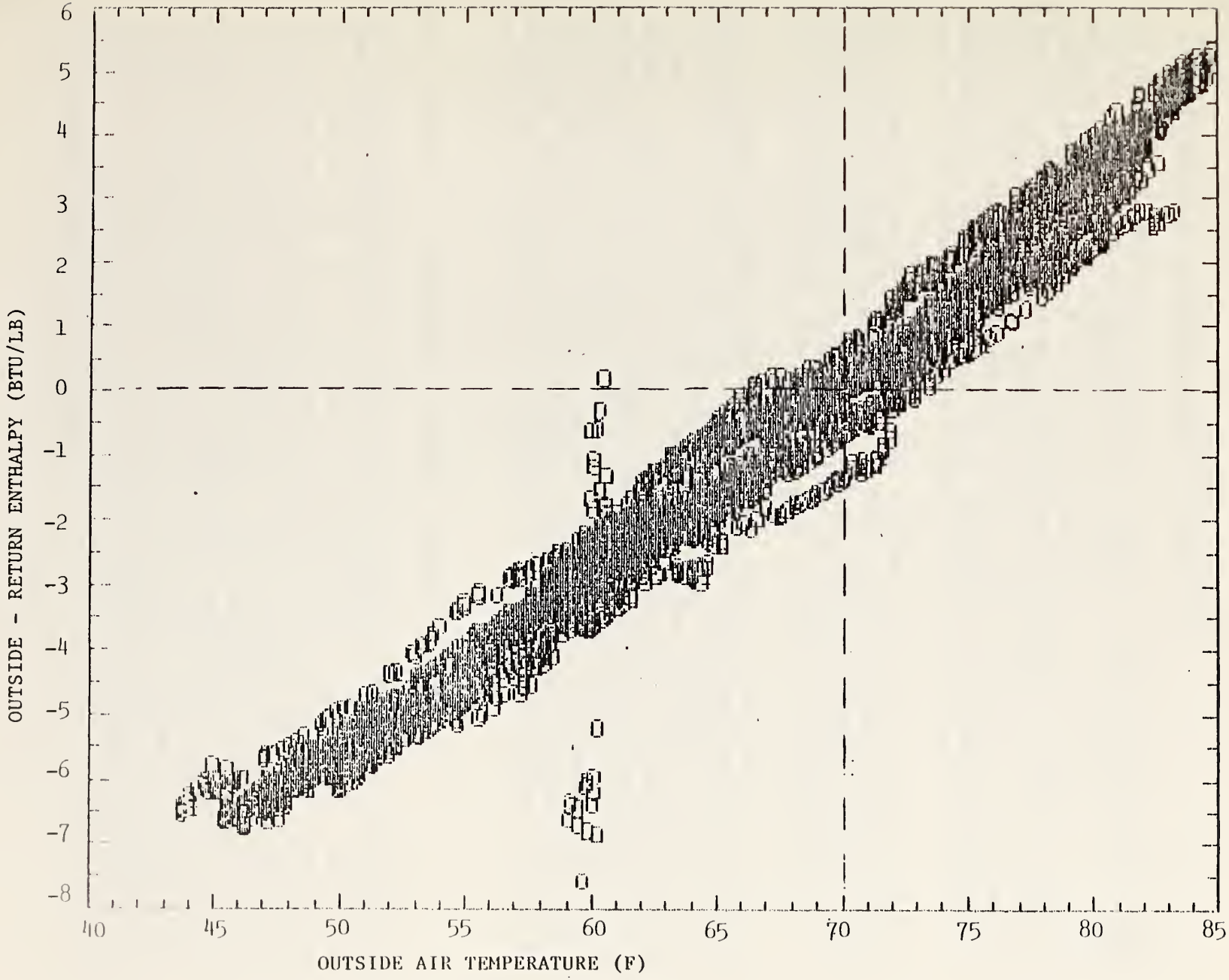

Figure 3-3. Enthalpy difference as a function of outside temperature (May 1-18, 1985)

4. VERIFICATION OF AN OPTIMUM START/STOP ALGORITHM

An optimum start/stop algorithm is used to reduce heating and cooling requirements and fan/pump electrical energy by turning off the HVAC system whenever it is not required. The criterion for whether or not space conditioning is required is usually whether or not the building is occupied. Then the optimum start/stop algorithm must determine the time to start the HVAC equipment so that a desired occupied space temperature is achieved just before the start of the occupied period. In addition, the algorithm must determine the time to stop the HVAC equipment so that the space temperature does not drift from the occupied setpoint until the building is unoccupied.

This algorithm can be applied to HVAC systems with central air handling units where the air handling unit is to be shut off during unoccupied hours or to systems with local zone heating equipment such as perimeter radiation where the local equipment is either shut off or the local space temperature setpoint for the local equipment control is changed (setback) for unoccupied hours.

All optimum start/stop algorithms perform the functions described above but the specific method used to calculate start and stop times can take many different forms. The complexity of the methods used varies from a simple linear correlation for start time as a function of indoor temperature and outdoor temperature with manually entered parameters to techniques involving modeling of the building and HVAC systems with adaptive determination of all parameters. A public domain optimum start/stop algorithm was developed at NBS and is described in a previous report [3]. The NBS public domain algorithm is intended to provide adaptive determination of parameters and uses two simple models for the behavior of a building zone when the HVAC system is on or off to predict the start and stop times.

4.1 Operation of the Optimum Start/Stop Algorithm

The public domain optimum start/stop algorithm was originally implemented in the language FORTRAN 77 in the form of a FORTRAN subroutine. The descriptions that follow will describe this implementation. The optimum start/stop algorithm subroutine is executed periodically at regular intervals. The reciprocal of the interval is the number of times per hour the optimum start/stop routine is called. This is the program variable NPTHR. The algorithm subroutine is called with three arguments which are the space temperature, the outdoor air temperature, and the time of day, which are obtained from the EMCS before calling the routine. The values of space and outdoor temperature are stored for a full day. Inside the algorithm a time scale is used which is different than the time of day. The internal time units are decimal hours, but the point at which the internal time (XTIME) is zero is the approximate midpoint between the start time and the stop time (near midday rather than midnight).

When the algorithm first starts, it must run for at least one day before any determination of the optimum start or stop time is made. The algorithm cannot be started at any time during the day. It must be started during the occupied period of the day. During the initial day of operation, the optimum start/stop algorithm turns off the HVAC system at the time of day when the building is to become unoccupied. The building space temperature is then monitored, and when the space temperature changes by a certain small value, the algorithm determines a value for the off-period dead time (TDEDOF) which is the difference in time between this point and the time the system was actually cut off. After the HVAC system has been shut off, the temperature of the space will drift upward or downward depending on the building and the weather. During the initialization phase, the algorithm does not attempt to calculate a start time. The time to start the system during initialization is an algorithm parameter. The initial start time should be early enough to start the HVAC system under any load conditions. After the HVAC system is started at the initial start time the algorithm monitors the space temperature. When the space temperature begins to move toward the setpoint, the algorithm calculates the on-period dead time (TEDEON) as the difference between the start time and the time at which the space temperature changes by a specified small value. When the space temperature reaches the occupied setpoint, the initialization is over and the algorithm program begins normal operation.

The optimum start/stop algorithm assumes that the variation of space temperature as a function of time can be described by a first order linear differential equation whose solution is an exponential function with a time constant and a steady state value. The parameters for any one exponential equation can be determined from two data points. The time constant for the equation is a function of the building structure and equipment capacity. The steady state value which the space temperature is assumed to approach after a long time depends on the outdoor air temperature and internal load. The behavior of the system after the HVAC system is started can be modeled by a start-up exponential equation, and the behavior of the system after the system is stopped can be modeled by a shut-down exponential equation. The intersection of these two equations is used to determine the equipment start time.

After each start-up, the program tries to determine the parameters for the start-up equation based on the behavior of the space temperature during the start-up period. The steady-state value is assumed to be 1.5 times the space setpoint temperature for heating, and 0.5 of the space setpoint temperature for cooling. The program determines the start-up time constant from two data points within the start-up period (at times X(1) and X(2)). These points are arbitrarily assumed to be located 1/3 and 2/3 of the way into the start-up period. Once the start-up equation parameters are determined, the space temperature can be predicted for any time during the start-up period. The algorithm assumes that the time constant for the next start-up period will be the same as the time constant for the previous start-up period.

After the start-up, the algorithm also recalculates the internal time coordinates (using the new start time), determines the minimum and maximum outdoor temperatures from the previous 24 hour period, and determines the time interval in which the next start time is assumed to be located.

During the occupied period after HVAC system start-up, the optimum start/stop algorithm only compiles outside temperature data. The stop time for the current day is always the shut-down dead time calculated from the previous day subtracted from the scheduled time at which the building becomes unoccupied. As the HVAC system is stopped each day, a new dead time is determined in the same way as during the initialization described above.

Once the HVAC system is turned off, the optimum start/stop algorithm must determine the next start time. In order to determine the start time, the algorithm must have an equation for the current shut-down period. As with the start-up equation, two parameters describe the equation and two data points are required to determine the parameters. The time for the first data point (stored in variable X(3)), is arbitrarily selected as the the internal time of day one quarter of the way between the time where the space temperature reacted to the previous shut-down and the time for the previous start-up. No start-time calculations are performed until this time is reached. When the time for the first data point (X(3)) is reached, the corresponding space temperature is stored (as Y(3)). After the first data point is determined, a determination of the shut-down equation parameters is made each time the algorithm is executed. The current space temperature is used for the second data point (Y(4)) along with the current time (X(4)). The two data points are used to determine the shut-down time constant which describes the behavior of the space temperature up to the current time. It is assumed that this value of the time constant can be used to determine space temperatures up to the occupancy time. The shut-down equation steady state temperature parameter or temperature at "infinity" time (TINF) depends on the trend of the space temperature since shut-down. If the space temperature is increasing, the previous day's maximum outside temperature is used as the TINF. If the temperature is decreasing, the previous day's minimum outside temperature is used as the TINF.

At each algorithm execution past the X(3) time, the algorithm has equations available to predict the shut-down and start-up space temperature behavior. The point in time for the optimum start is where the two equations intersect. A special subroutine (ROOT) is used to find the intersection of the two equations. The interval to search for an intersection was previously calculated. Both the start-up and shut-down curves will each have two points corresponding to where they cross into and out of this search interval. These intersection points can be used to determine if the equations cross within the interval. If both limit intersection points for one curve lie above the limit intersection points for the other, there is no intersection. The root subroutine will be unable to produce a solution. Otherwise, the routine will determine the crossing point. If the two curves are very similar in slope, (or

if one slope is very large) the routine may require a large number of iterations to find the answer.

Once the intersection of the unoccupied and start-up equations is determined, this intersection can be compared to the current time. If the times are the same or the intersection time has passed, the HVAC system is started. After the space temperature has responded, a new start-up dead time is calculated and the cycle of algorithm calculations begins again.

4.2 Installation of Algorithm

The public domain optimum start/stop algorithm was installed on the NBS laboratory EMCS. The algorithm was installed on the central control computer rather than on the field processor. Since the original algorithm implementation was written in FORTRAN 77 and the field processor software was largely written in FORTRAN IV, installing the algorithm in the central computer avoided problems with translation. A loss of communication between the central computer and the field processor would have resulted in the optimum start/stop algorithm being unable to function. However, a scheduled start/stop algorithm installed in the field processor ensured that a start and stop would take place at a scheduled time if the optimum start/stop algorithm did not issue a start or stop command. The placement of the algorithm at the central level also allows coordinated start/stop control of all HVAC equipment.

A single air handling unit (described in the introduction) was used as the HVAC equipment to be started and stopped by the optimum start/stop algorithm. The air handling unit fan control start and stop relays were connected to digital outputs under the control of the EMCS field processor. Application software in the field processor was used to perform the sequence of operations necessary to start up and shut down the air handling unit. The shut down consisted of stopping the supply and return fan, shutting all dampers and valves, and disabling direct digital control of valves and dampers. Start up consisted of starting the supply and return fans and enabling direct digital control of valves and dampers.

Two special support subroutines were used with the optimum start/stop algorithm. One of these was used to obtain the current outdoor and space temperatures from the field processor. The space temperature sensor was located in one of the 39 offices supplied with air by the air handling unit. This office was assumed to be representative of all of the offices. The outside air temperature sensor was located just within the outside air dampers for the air handling unit. The disadvantage of this location is that during the period when the air handling unit was not operating, this temperature sensor was not directly exposed to outside air since the dampers were closed and there was no air flow through the dampers. Since minimum outside air temperatures usually occur during unoccupied hours, it is likely the minimum

outside air temperatures used during the optimum start/stop algorithm tests were not strictly correct.

Another special support subroutine used with the optimum/start stop algorithm provided the algorithm the ability to start and stop the air handling unit. The routine sent a command to the field processor to start the software which started up or shut down the air handling unit. The software had safety features installed to prevent the unit from being started or stopped within one minute of having been previously stopped or started.

The main program described in reference [3] was designed for testing the optimum start/stop algorithm in a programming environment. This routine was modified to operate in an EMCS environment. The original program contained a programming loop which endlessly read outdoor and space temperatures from a file and executed the optimum start/stop algorithm. A delay was placed in this loop to restrict execution of the algorithm to regular intervals as specified in the algorithm parameters. Temperatures were obtained directly from the EMCS rather than a file. An output file was used to write messages on the status of the algorithm and values of calculated start and stop times.

A file was also used to load all algorithm parameters when the algorithm was started. The parameter file contained the parameters listed in table 4-1. Many of the parameters in the table are easily determined such as the identification number for the space temperature sensor, the identification number for the outside air temperature sensor, the identification number of field processor, the time of day the building is occupied, the time of day the building becomes unoccupied, the initial start-up time, and the initial season mode (heating season or cooling season). The assignment of values to other parameters is more complicated. These parameters are defined and discussed in section 4.5.1.

4.3 Corrections to Algorithm

As the public domain optimum start/stop algorithm was studied during the course of installation and testing, a number of problems were uncovered. These problems were either due to errors in the computer program or to deficiencies in the original implementation of the algorithm which caused the algorithm to fail in unforeseen situations. Corrections and additions were made to the implementation of the algorithm until the algorithm was able to run properly in control of an actual air handling unit. The following sections describe the modifications which were made to the algorithm.

4.3.1 Conversion of Internal Constants to Externally Assigned Parameters

The optimum start/stop algorithm, as originally implemented, had a number of parameters which required recompilation of the computer program to change. These parameters were placed in FORTRAN common areas and assigned values which were read from a parameter file on program initialization. The parameters which were previously internally assigned included the sample (execution)

frequency for the algorithm and a number of parameters used for mathematical comparisons in the program. The parameters read from the parameter file are described in section 4.5.1.

table 4-1 optimum start/stop algorithm parameters

parameter	used in test
1. number of times per hour for the algorithm to execute.	12
2. identification number for space temperature sensor.	11016
3. identification number for outside air temperature sensor.	11011
4. identification number of field processor.	2
5. time of day building becomes occupied. (decimal hours)	7.75
6. time of day building becomes unoccupied. (decimal hours)	18.00
7. initial start up time.	6.00
8. space temperature setpoint. (F)	73.00
9. minimum desired space temperature. (F)	50.00
10. maximum desired space temperature. (F)	90.00
11. temperature differential around maximum or minimum space temperature to prevent oscillation when limits of space temperature are exceeded. (F)	1.00
12. initial season mode, heating season or cooling season.	heating
13. minimum differential for intersection of start-up and shut-down curves in root finding subroutine. (F)	0.10
14. differential for determining when the time for the first data point for calculation of the shut-down time constant has been reached. (F)	0.06
15. change in space temperature to define shut-down dead time. (F)	0.25
16. change in space temperature to define start-up dead time. (F)	0.25
17. allowed variation in space temperature around the space setpoint. (F)	5.00
18. earliest time of day that HVAC system can be turned off. (hrs)	17.167

4.3.2 Problems with Calculation of Time Constants

During testing of the optimum start/stop algorithm it was observed that under certain conditions the mathematical computation of the constants for the start-up or shut-down equations would involve division by zero or taking the logarithm of a number less than or equal to zero. To prevent algorithm failures, it was necessary to add insert tests of the variables involved to insure that an arithmetic fault would not be generated. The time constants are computed in the FORTRAN functions YONNEW and YOFF and this is where the tests were inserted. A FORTRAN parameter (EPSTAU) was used as a lower limit for the value of the time constant. If the time constant was less than this value, it was set to the lower limit. Changes were also made to cause the algorithm to output warning messages if either of the time constants were computed as zero.

4.3.3 Determination of previous unoccupied period maximum and minimum outdoor air temperatures.

The original algorithm, as published, had an error on the initialization phase of the algorithm. At each system start-up, the algorithm determines the maximum and minimum outdoor air temperatures from the unoccupied period just ending. A subroutine (MAXMIN) determines these values. This subroutine searched the previously stored data from the start of the unoccupied period on the preceding day to the start of the current occupied period. These search limits were fixed within the subroutine. During the initial cycle of the algorithm, or the first time this maximum/minimum search is made, there is no existing data between the start-up time and the occupancy time. This section of the stored data area is undefined. The undefined area is included in the search and therefore the maximum and minimum temperatures determined could be incorrect. To correct this problem, the MAXMIN subroutine was altered to include as subroutine arguments the search limits for locating the maximum and minimum values within the temperature storage array. The first time the MAXMIN routine is called the search limits are between the start of the occupied period on the previous day to the current time of day.

4.3.4 Problems with HVAC Equipment Failing to Obey Start/Stop Commands

The optimum start/stop algorithm, as originally written, used a logical variable to control the HVAC equipment under start/stop control. It was assumed that when the logical variable was set to true the equipment would start and when the variable was set to false the equipment would stop. In a real EMCS it is to be expected that equipment will not always obey commands, due to communication failure, power failure, or HVAC equipment failure. Most EMCS have a method of verifying that when a command is issued the equipment actually responds. An alarm of some sort is usually generated when the equipment does not respond.

In the original optimum start/stop algorithm, if the equipment did not respond the algorithm would still assume that the equipment was going to respond. This can result in gross errors in the start and stop time calculations. An example of this would be if a command was issued by the algorithm to start an air handling unit and for some reason the unit did not start. The algorithm is then waiting for the space temperature to begin to change. Since the unit did not start, the space temperature will not change. Eventually it will be noticed that the unit did not start and it may be manually started. The manual start will cause the space temperature to change, but the change may have occurred much later than it might have. This will cause the start-up dead time to be excessively large, causing an error in the start-time calculated for the next day.

The original optimum start/stop algorithm was modified to allow it to respond properly to situations where HVAC equipment fails to follow commands from the algorithm. A logical variable ALARM was added to the software and placed in a

FORTRAN common area where it is assumed that the EMCS software will set the variable to a true state if the HVAC equipment fails to respond to command. If the ALARM variable is in the true condition and the algorithm is waiting for the space temperature to react following a start or stop action, then the algorithm will reset internal control variables to prevent internal variables from being recalculated and will set internal status variables to the correct state. Calculations for the start and stop times will then use the values of the internal variables for the previous day. These changes were made in the main algorithm subroutine, OPTSS.

4.3.5 Improvement to Allow Restart of Algorithm without Initialization

The optimum start/stop algorithm, since it is adaptive, requires an initialization period to determine initial values for the internal variables used to calculate start and stop times. Once the initial values of internal variables are determined, modifications are made to the variables each day to adapt to changing conditions in the weather and the building. If it were necessary to stop and restart the algorithm, such as during maintenance or failure of the EMCS computer, then the algorithm as originally written would have to go through the two-day initialization period after it was re-started. This would result in the loss of potential energy savings.

To avoid this, the optimum start/stop algorithm was modified so that critical internal variables would be written to a storage file once each day. Another modification was made to cause the algorithm to try and find a critical variable initialization file when the algorithm was restarted and if such a file was found, read values for the internal variables from the file and bypass the initialization period. These modifications made the testing of the algorithm easier since it was not always necessary to reinitialize after a test or after any interruption of a testing period.

4.3.6 Preventing the Setback Controller from Operating During Occupancy

The optimum start/stop algorithm includes a controller (on-off controller) to maintain a minimum (or maximum) temperature in the building space during unoccupied hours by starting and stopping the HVAC equipment if the space temperature exceeds limits. During testing of the optimum start/stop algorithm it was determined that under certain unusual circumstances the on-off controller became active during occupied hours. Additions were made to the main algorithm subroutine to prevent the on-off controller from becoming active during occupied hours.

4.3.7 Limitation to Prevent Excessively Early Stop Time

During the spring and fall when the space temperature does not change very much after the HVAC equipment is shut down, the optimum start/stop algorithm

may determine a shut-down dead time which is very large. In this case, the next day the HVAC equipment may be shut down very early. The space temperature may not degrade to uncomfortable levels before the end of the occupied period, but the air quality may become unacceptable. In order to avoid this situation, an additional parameter was added to the optimum start/stop algorithm to limit the stop time from being excessively early. If the calculated stop time is before the time specified by this parameter, which is read from a parameter file, then the stop time is set to this stop time limit. This change was made in the main subroutine of the optimum start/stop algorithm, OPTSS.

4.3.8 Automatic Switching between Heating and Cooling Modes

The public domain optimum start/stop algorithm was originally designed to operate in a defined season. The heating season was defined as the time of year when the space temperature drops after HVAC system shut down and the cooling season was defined as the time of year when the space temperature climbs after shut down. When the algorithm, as originally written, is exposed to situations where the space temperature behavior after shut down is not strongly rising or falling, the start time becomes difficult to determine. This is due to the following two types of problems.

When the weather conditions that a building is exposed to become mild, the space temperature in the building tends to remain constant after system shut down. When the HVAC equipment is started up, the start up time before building occupancy is very small since the space temperature is already near the setpoint. In the extreme case, on a plot of space temperature versus time, the start up curve becomes a near vertical line at the occupancy time and shut down curve becomes a horizontal line at the space setpoint. The subroutine to determine the intersection (ROOT) will then require a large number of iterations to find the intersection. The maximum number of iterations in the original algorithm was too small and this maximum was increased. When the maximum number of iterations is reached without a solution, the root is assumed to be at the leftmost boundary of the previously determined search limits for finding the root.

If the optimum start/stop algorithm is in heating mode and the weather becomes warm, it is possible that the start-up and shut-down equations may not have an intersection within the search interval. In the original algorithm, the root would then be selected as the leftmost boundary of the previously determined search limits for finding the root. Additions were made to the algorithm to allow the algorithm to respond in a different manner to this situation. After the changes, whenever the root subroutine is unable to find a root in the interval, the mode of the algorithm is changed from heating to cooling or cooling to heating and a second attempt is made to find the root. The reason that this works is that the start-up equation parameters depend on the mode of the algorithm. The steady-state temperature for the start-up equation is 0.5 multiplied by the space setpoint for cooling and 1.5 multiplied by the space setpoint for heating. By switching the mode of the equation, the start-up

curve is approximately rotated about the space temperature setpoint, resulting in an intersection of the start-up and shut-down curves for the new algorithm mode. This change to the optimum start/stop algorithm results in it being unecessary to manually change the season mode of the algorithm. Changes were made in the main optimum start/stop algorithm subroutine, OPTSS, and in the root finding subroutine, ROOT.

4.4 Testing of Algorithm

Testing of the public domain optimum start/stop algorithm was begun on July 27, 1984, using a building emulator rather than an actual building. The building emulator is a computer connected to the NBS laboratory EMCS which uses a simulation computer program to control signals on analog and digital inputs to the EMCS in response to control signals from the EMCS. The emulator was designed to emulate the response of a real air handling unit in a building. Use of the emulator had the advantage that any problems with the algorithm did not affect actual building occupants, and weather and occupancy conditions could be completely controlled. Testing using the emulator pointed out most of the problems described in section 4.3, and continued until November 21, 1984.

The optimum start/stop algorithm was tested on an actual building air handling unit for a period starting on September 20, 1984. Some problems were observed during the acutal building tests. All modifications to the original optimum start/stop algorithm were completed by October 2, 1984. Testing took place on the completed algorithm during the period from October 2 to December 10, 1984, with only one interruption due to a plant outage. The parameters used in testing on the actual building are listed in table 4-1.

4.4.1 Test Procedure

One assumption made during the development of the public domain optimum start/stop algorithm was that the space temperature in a building with equipment controlled by the algorithm remains at the space temperature setpoint throughout the occupied period. When the algorithm was tested on the actual building, it was observed that the space temperature drifted between 68.5 and 77 F, depending on the time of day and weather conditions. Section 9.4.1 describes the local control equipment in the building used for testing. This local equipment did not operate in an ideal manner and did not maintain the space at a fixed setpoint. Another observation about the behavior of the space temperature is that the temperature in the room would change by two or three degrees F if the door to the office were closed rather than open. Closing the door caused the amount of air flow through the office to change. In order to test the optimum start/stop algorithm a setpoint of 73 F was used with a allowance of 5 F on either side of the setpoint as being acceptable.

Another assumption made in developing the optimum start/stop algorithm was that the space temperature will change appreciably during the unoccupied period, at least during the winter or summer. The space temperature did not drop very low during unoccupied hours with the air handling unit was off throughout the testing period, even with outside air temperatures of 25 to 30 F. This was probably due to two major reasons. First, the offices in the building contain a reheat coil which still operated when the air handling unit was off. This would prevent the space temperatures from dropping very low. The reheat coil setpoint was manually controlled and could not be setback during unoccupied hours. A second reason is that the air handling unit under control was not the only unit in the building, and at least one other air handling unit was left running throughout the unoccupied period to provide essential ventilation. This meant that the space used for testing the optimum start/stop algorithm was indirectly conditioned, even when the test air handling unit was off.

For purposes of testing the optimum start/stop algorithm, data were collected from the air handling unit and from the optimum start/stop algorithm output for the period when the algorithm was controlling the air handling unit between October 2 and December 12, 1985.

4.4.2 Test Results

During the test period, recordings were made of the temperatures and actuator positions in the building space and air handling unit every five minutes. In addition, the optimum start/stop algorithm output information describing results of calculations and timing of start/stop events was saved. A compilation was made each day of several important criteria used in evaluating the optimum start/stop algorithm's performance. These were the number of iterations required to find the intersection of the start-up and shut-down curves for the start time, the calculated start time, the difference between the actual and setpoint space temperature at the time of building occupancy, the difference between the occupancy time and the time that the building space reached the setpoint, the calculated stop time, the difference between the actual and setpoint space temperatures at the time the building was unoccupied, and the maximum and minimum measured outdoor air temperatures for the day.

During the testing period, the algorithm in general operated properly, with a few instances where a start or stop time was unusual. Figure 4-1 is a plot of the calculated start and stop times over the testing period. The upper curve is the stop time, which always lies slightly below the time when the building was considered unoccupied at 18:00. The lower curve is the calculated start time which, with two exceptions, lies below the time when the building was considered occupied at 7:45, which is marked with the dashed line. The first early start time is during the initialization of the algorithm. The second early start time is at 3:00. This occurred after an instance where the start of the HVAC system on the previous day did not cause the space temperature to

change appreciably for 4.5 hours after start-up. This resulted in a large calculated start-up dead time. The algorithm assumed that the same dead-time was valid for the next day and started the system 4.5 hours early. The algorithm was able to recover by the next day however. At the end of the test period the algorithm also started the system early but was not able to recover and began to start the system progressively earlier each day until the algorithm was stopped manually. This data is not shown in figure 4-1.

Figure 4-2 is the same start time data shown in figure 4-1 plotted versus the minimum outside air temperature measured during previous unoccupied period. There is no clear trend visible in the data, illustrating that the building space temperature behavior does not appear to be sensitive to the outside air temperature.

Two criteria were used to determine the performance of the algorithm in selecting a reasonable start time. The first of these is the difference between the occupancy time and the time that the space temperature actually arrived at the setpoint. Figure 4-3 is a histogram of the number of occurrences of various magnitudes of this difference using 5 minute bins. A negative number for the difference indicates that the space temperature reached the setpoint before the occupancy time. This figure shows that much of the time the algorithm was within 5 minutes of the correct start time. Occasionally the setpoint was reached fairly late.

A different way to present the start time performance of the algorithm is by using the criterion of the difference between the space temperature and the space temperature setpoint at the time of building occupancy. Figure 4-4 is a histogram of this difference for start-ups during the test period using 0.25 F bins. A negative value of error indicates a space temperature below the setpoint of 73 F. This plot shows that in only one case was the space temperature out of the allowable limits of 68 to 78 F during the testing, but was usually one to four degrees F low.

The stop time selection performance of the optimum start/stop algorithm is shown in figure 4-5. This figure is similar to figure 4-4 but shows a histogram of the difference between the space temperature and the setpoint at the time when the building is unoccupied. The error appears to be evenly distributed around the zero error point with all errors within 4 F of the setpoint.

4.5 Considerations in Use of the Optimum Start/Stop Algorithm

In general the test results show that the public domain start/stop algorithm does operate, but is subject to occasional failures. Further improvements to the algorithm would probably remove such instances. The building used and the test conditions to which the algorithm were exposed were certainly not adequate to test the algorithm completely. The testing did show that the algorithm was able to function in this limited case however and shows promise

of determining the optimum start and stop times in other buildings. The following sections discuss any conclusions that were reached on the selection of values for optimum start/stop algorithm parameters and general usage of the algorithm beyond any recommendations given in reference [3].

4.5.1 Parameter Selection

The optimum start/stop algorithm has a large number of parameters which must be assigned values. Fortunately, since the algorithm is theoretically adaptive, it is not necessary to assign values to parameters which describe the building and the HVAC equipment in the building. The parameters which must be given values determine how stable and how accurate the algorithm will be.

In the descriptions below, an epsilon is defined as a small number which is used to determine if two numbers are close enough to be considered the same. Several of these are used in the algorithm.

4.5.1.1 Selection of algorithm sampling rate

The algorithm sampling rate is the number of times per hour that the optimum start/stop algorithm will execute. The most important criterion for selection of the sampling rate is the time the building takes after start-up to reach the setpoint. This time will be a maximum in the winter and summer and can conceivably become zero in the spring or fall. In order for the optimum start/stop algorithm to operate properly, there must be a least four space temperature samples within the start-up period. If there are less than three samples, then the time constant calculated for start-up will be zero and the building will be started at the occupancy time. At some times of the year as winter or summer is approached, starting the building at the occupancy time will be unacceptable. The time required to start-up at this time of year should have at least four data points within in it. For example, if the start-up time before occupancy is 15 minutes, then the sample interval should be less than or equal to five minutes and the sampling rate would be 12 times per hour.

Another less important consideration in selection of the sampling rate is the minimum acceptable resolution of the start or stop time. For example, during the testing of the algorithm, a sampling rate of 12 times per hour was used. The algorithm executed every five minutes and therefore the start and stop times could be selected only to the nearest five minutes. This means that if the algorithm does not pick exactly the correct start time, it will be in error by five minutes, ten minutes, or other multiples of five minutes. If too great a rate is used, the EMCS will be slowed by excessive processing.

4.5.1.2 Selection of epsilon for ROOT subroutine

The ROOT subroutine is used to determine the optimum start time by finding the intersection of the start up and shut down equations. The subroutine will go through a series of iterations until a point in time is found where the temperature predicted by the start-up equation and the temperature predicted by the shut down equation are sufficiently close. The minimum closeness required is the epsilon for the ROOT subroutine. If this value is too small, an excessive number of iterations will be required to find the intersection. If the epsilon is too large, this will limit the accuracy of the start time determination.

4.5.1.3 Selection of epsilon for shut-down dead time

The epsilon for shut-down dead time is the amount of change in the space temperature that is allowable after the HVAC equipment is shut down while the building is still occupied. The shut-down dead time is the time required for the space temperature to change by the amount of the epsilon after shut down. If the value selected is too small, the potential savings from optimum stop of HVAC equipment may be limited. If the value is too large, then occupant comfort at the end of the occupied period may suffer.

4.5.1.4 Selection of epsilon for start-up dead time

The epsilon for start-up dead time is used to determine the amount of time after HVAC system start up that is required before the space temperature begins to respond. It is defined as the change in space temperature that must occur before the dead time is calculated. If this value is too large, then calculated dead times are likely to be excessive, resulting in earlier starts than necessary. If the value is too small, calculated dead times may be unrealistically small, causing start times which are later than they should be.

4.5.1.5 Selection of epsilon for Y(3) vs. X(3) determination

The epsilon for determining whether or not the time for the first data point for the shut down equation (X(3)) has been reached is defined as the interval of time in decimal hours away from the X(3) point within which the current time must be for the algorithm to take the current value of space temperature as the value to use in figuring the shut down equation. If this epsilon is too small, the X(3),Y(3) point can be passed without detection by the algorithm. If the epsilon is too large, the value of space temperature used for the data point could be, for example, a time that is several minutes away from the correct X(3) time value. The proper value for this epsilon should be larger than half of the sampling time interval but less than the

largest time interval that would result in an excessively large deviation of the space temperature Y(3) from the space temperature exactly at time X(3).

4.5.1.6 Selection of minimum and maximum space temperatures

The minimum and maximum space temperature parameters are used for two purposes. The optimum start/stop algorithm contains a controller to maintain the space temperature during unoccupied hours within some limits. If the space temperature during the unoccupied period is higher than the maximum space temperature or lower than the minimum space temperature, then the setback or on-off controller becomes active, turning on the HVAC equipment until the space temperature is within the minimum or maximum temperature by another parameter (DELY). The minimum and maximum temperatures are also used in the calculation of the time interval within which the ROOT subroutine looks for the intersection of the start up and shut down equations. The values selected should be equal to the maximum and minimum temperatures that will avoid problems in the building space due to condensation, overheating, freezing or similar difficulties.

4.5.1.7 Selection of epsilon for tolerance in space temperature setpoint

The epsilon for the space temperature setpoint is used to determine the end of the start up period. When the difference in temperature between the current temperature and the space setpoint is less than this epsilon, the space temperature is considered to be within the limits acceptable for the occupied period. If the building in which the optimum start/stop algorithm is being operated has local zone controls which keep the space temperature at a constant value within a small tolerance, then the space temperature setpoint epsilon can be small. If the space temperature drifts between wide limits, then the epsilon should be chosen to correspond to the limits of drift.

4.5.1.8 Selection of earliest allowable shut-down time

Under conditions of light loading or mild weather, HVAC equipment may be shut down in a building and the space temperature will remain constant or will not change for hours. On the basis of space temperature, the optimum stop time could then be hours before the end of occupancy. However, if the building relies on the HVAC equipment for ventilation air, or moisture removal, premature shut down could cause occupant discomfort. The earliest allowable shut down time is a parameter which is used to prevent the optimum start/stop algorithm from shutting down the HVAC equipment before a certain time-of-day after which no occupant discomfort will result due to insufficient ventilation or odor/moisture removal. This value may be estimated but will most likely have to be determined by trial and error.

4.5.2 Constraints to Usage of Algorithm

The public domain optimum start/stop algorithm, as modified from the original algorithm described in reference [3], will determine the optimum start and stop times for HVAC equipment. The optimum start and stop times are chosen on the basis of space temperature alone, and factors, such as ventilation, removal of space contaminants, and space relative humidity, are not directly taken into account. Based on the limited testing of the algorithm discussed in this chapter, it appears that the algorithm can be expected to fail under some unusual conditions, where failure is defined as either prediction of start or stop times which are grossly wrong or not causing any start or stop actions at all. Further work is necessary to test the algorithm under a wider variety of conditions to determine the circumstances under which the algorithm may fail and modify the algorithm to prevent failures. The testing described in this report has shown, however, that the algorithm does basically perform as expected in predicting start and stop times.

4.6 Revised Optimal Start/Stop Algorithms

Because many changes were made in the original optimal start/stop algorithm, a FORTRAN 77 listing of the revised algorithm, as implemented on the NBS Laboratory EMCS, is included in Appendix A. The changes made, most of which have been discussed in this report, are summarized in comment statements at the beginning of the program. In addition, modifications to actual lines of code are indicated with a comment showing when the addition, deletion or change was made.

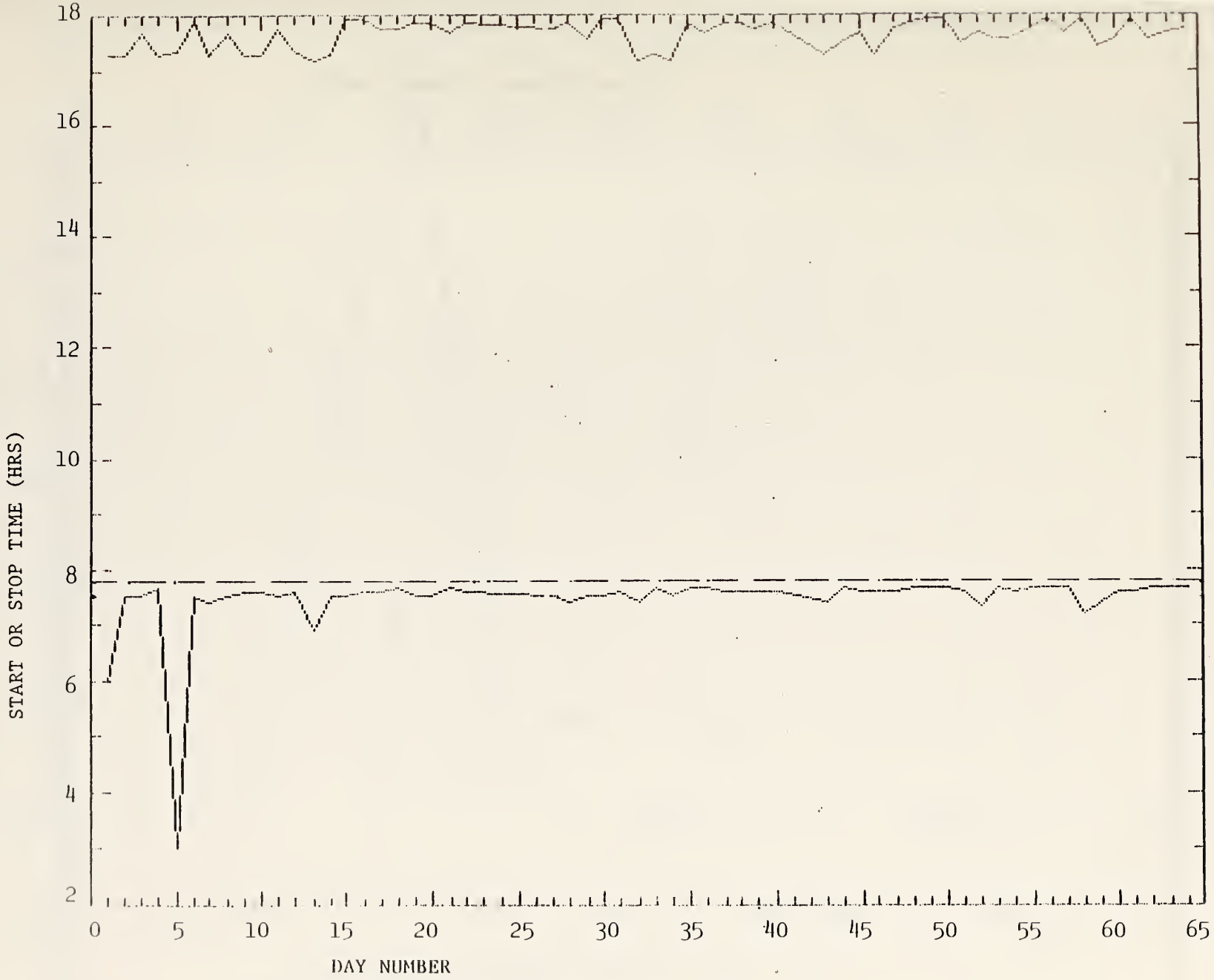

Figure 4-1. Optimum start/stop algorithm calculated start and stop times

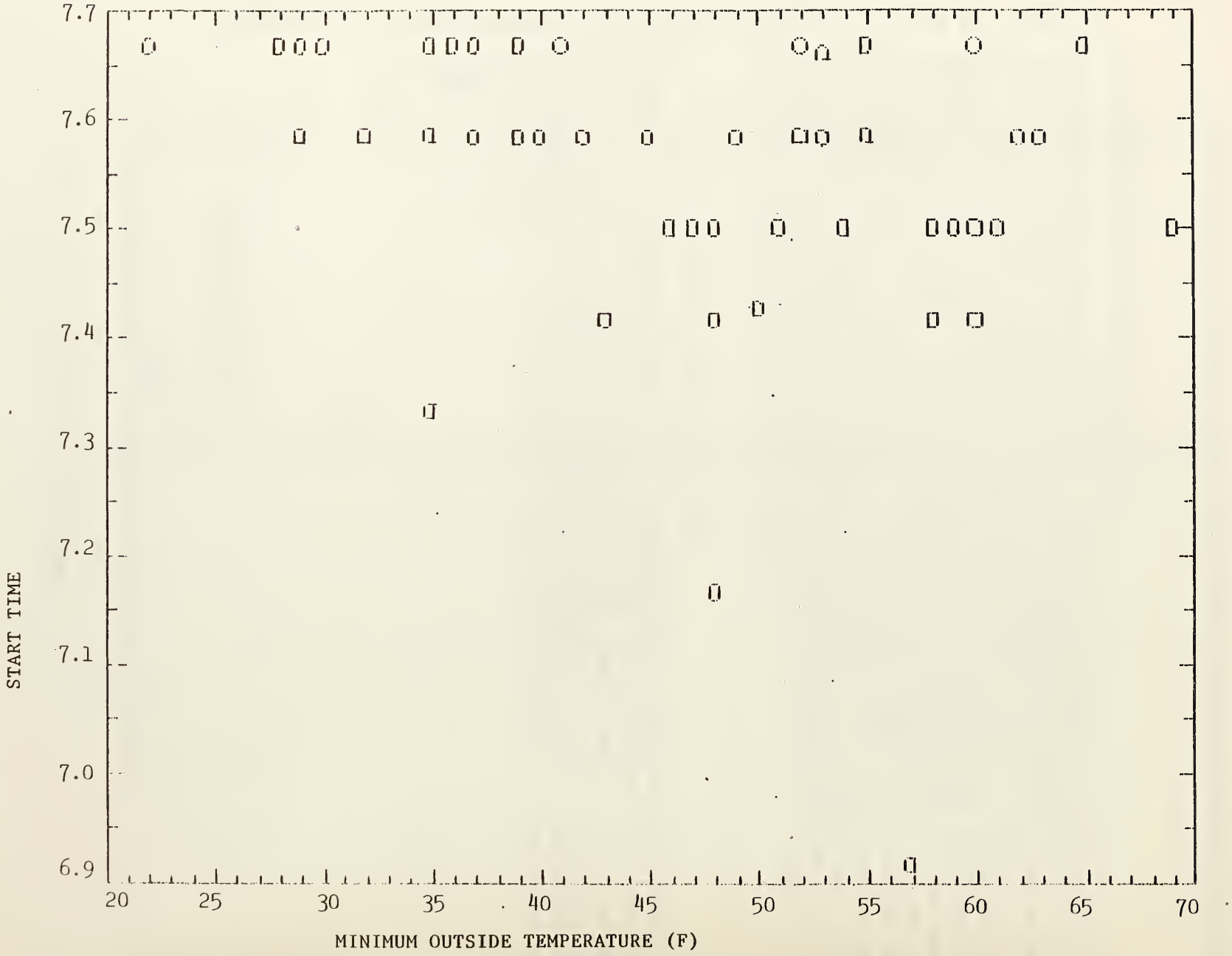

Figure 4-2. Calculated optimum start times versus minimum outside temperature

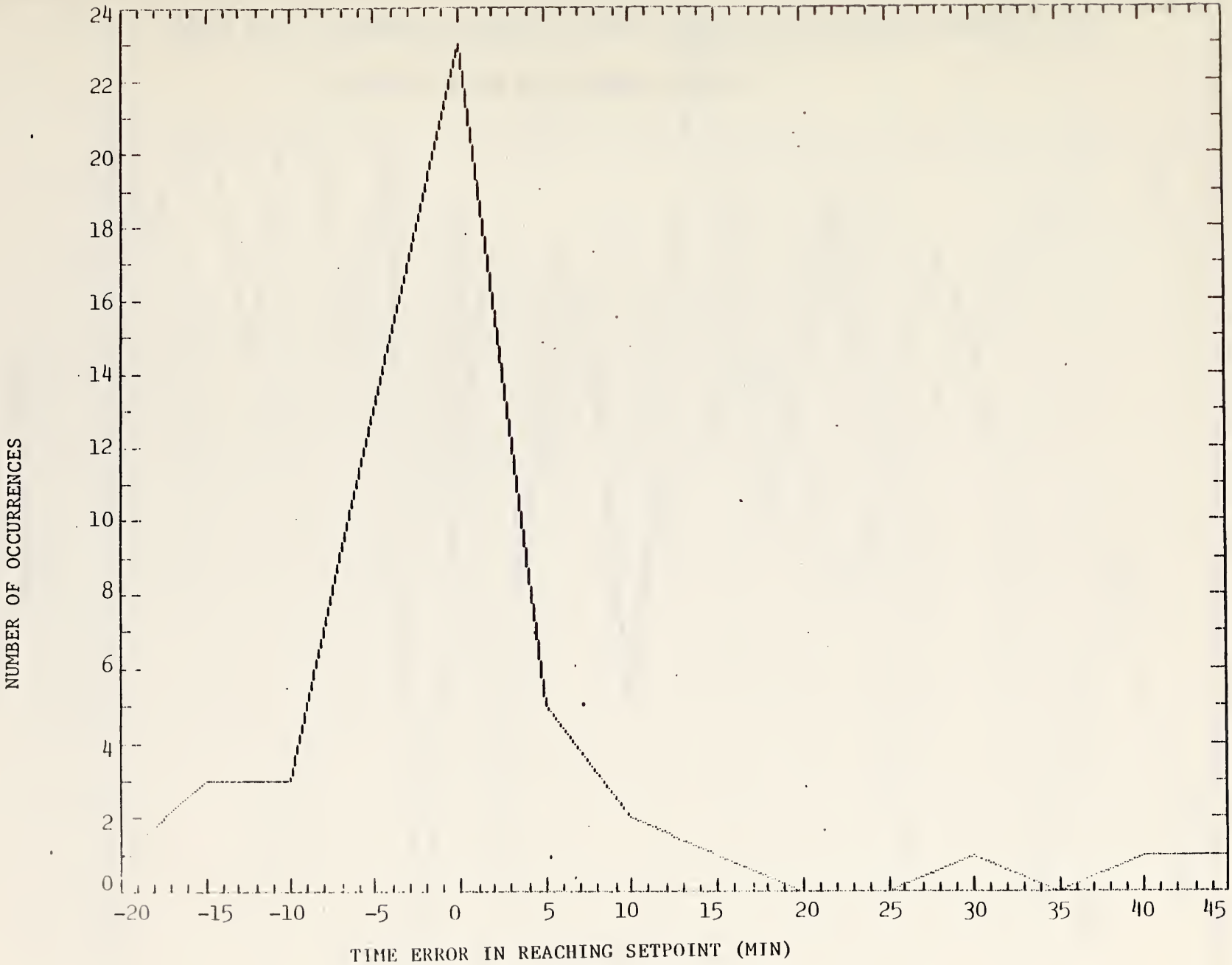

Figure 4.3. Histogram of error in time between occupancy time and arrival at setpoint

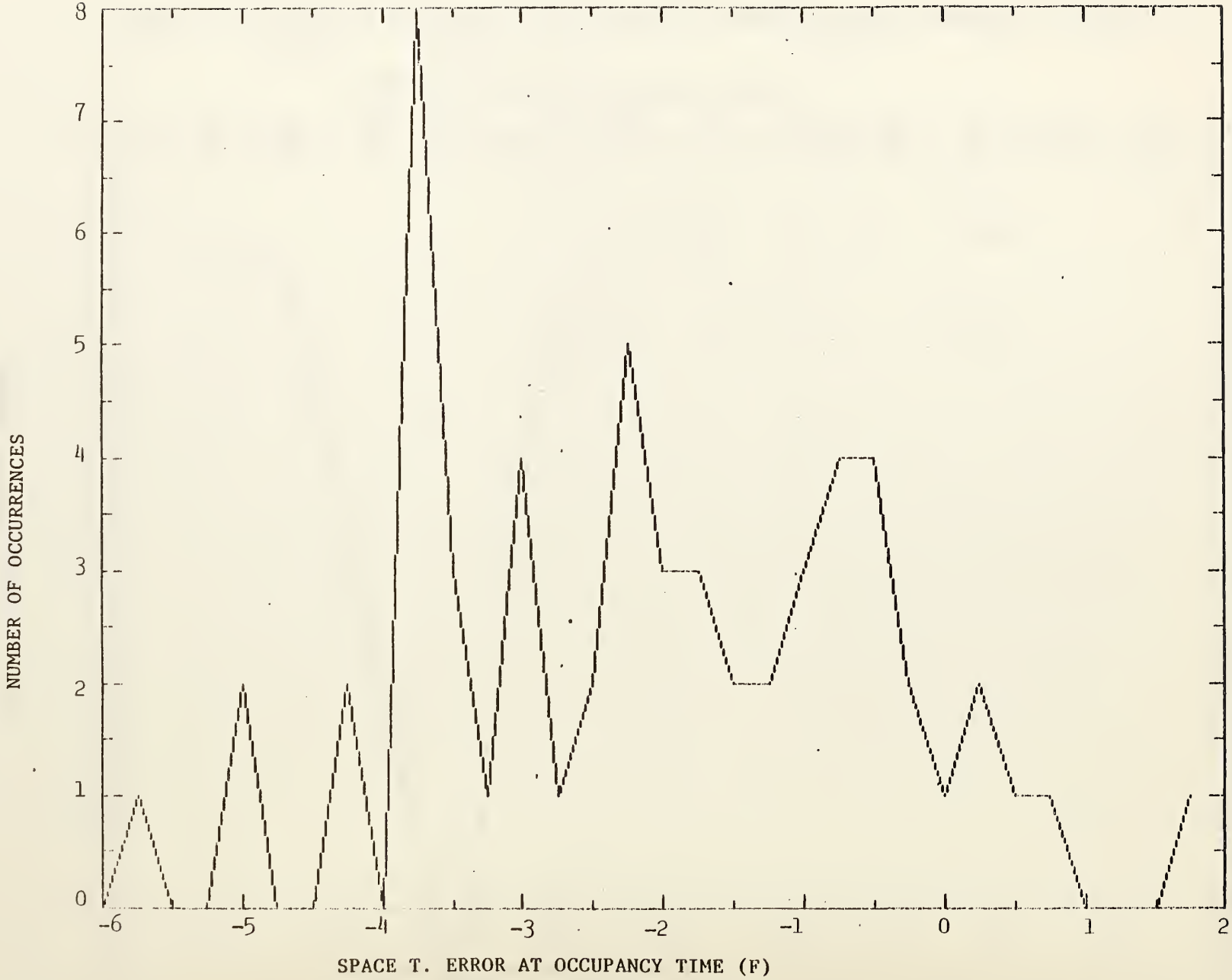

Figure 4-4. Histogram of error in space temperature at building occupancy time

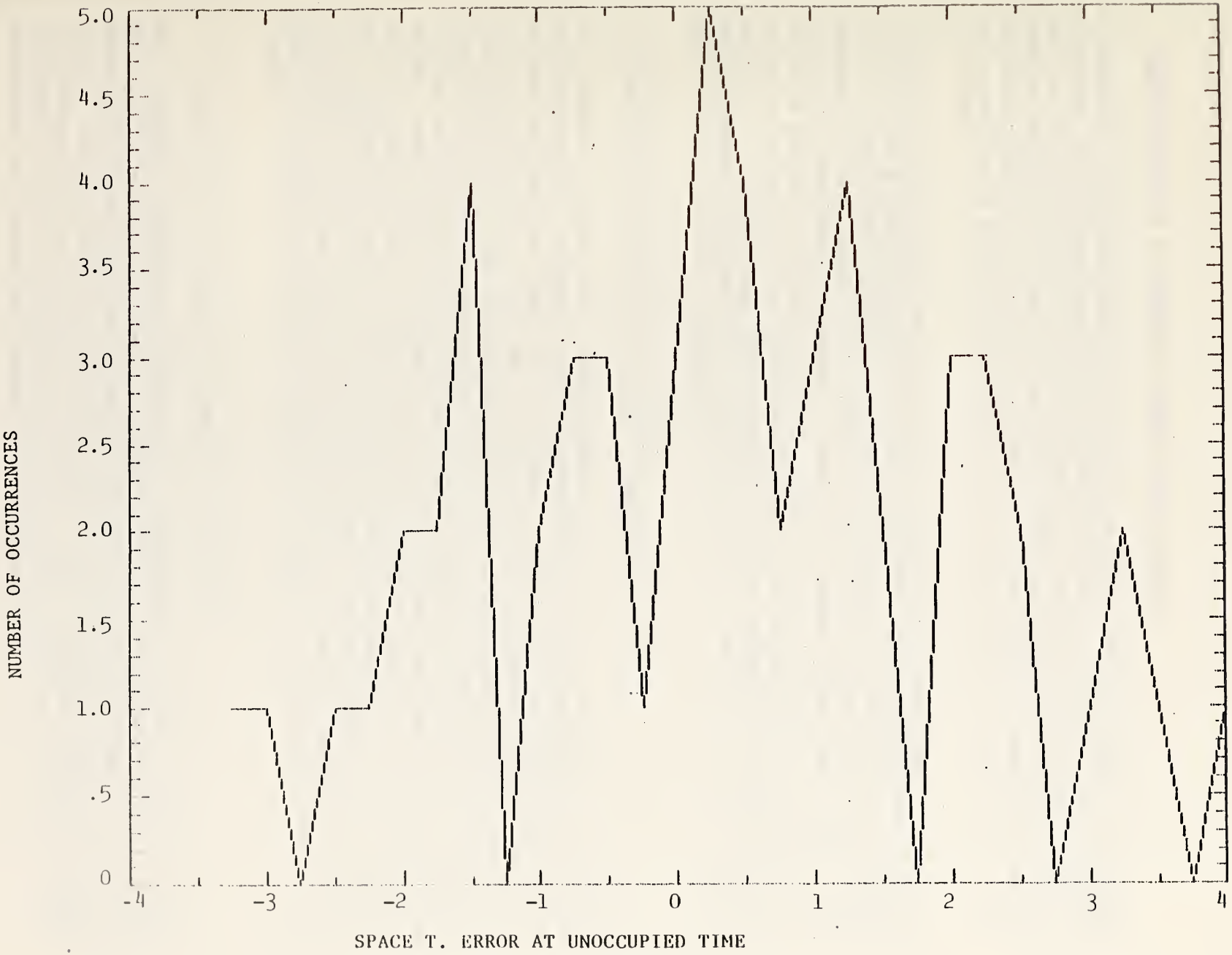

Figure 4-5. Histogram of error in space temperature at time building is unoccupied

5. VERIFICATION OF A SCHEDULED START/STOP ALGORITHM

A scheduled start/stop algorithm is used to start and stop specific pieces of equipment or lighting loads at predetermined times of day. Such an algorithm must not only be able to schedule control of loads, but must also be able to resolve conflicts with other algorithms which may control the same loads. The scheduled start/stop algorithm may contain its own conflict resolution section or it may use a lower level load control algorithm in common with other algorithms.

A public domain scheduled start/stop algorithm was developed at NBS and is described in a previous report [2]. The algorithm in that report was referred to as a time-of-day control algorithm, which is intended to be a broader category of algorithms than scheduled start/stop algorithms. Time-of-day control is considered to extend to control of events rather than loads, where an event may be a single execution of an algorithm, control of a series of loads, or starting a periodic execution of an algorithm, as well as the control of a single load. The following sections describe the testing of the public domain scheduled start/stop algorithm for the control of the stopping and starting of an air handling unit.

5.1 Operation of Scheduled Start/Stop Algorithm

The operation of the public domain scheduled start/stop algorithm is described in reference [2]. However, a brief description of the algorithm will be given in this section. The algorithm is assumed to have access to information which has been previously stored about loads or events to be controlled, the day of week and time of day to control the event or load, and the nature of the control (for example stop, start, cycle once). The algorithm is assumed to execute periodically and at each execution examines the control time for each of the loads which are scheduled. If the difference between the current time and the control time is within a tolerance (such as one minute), the algorithm will initiate the event. The control over events is assumed to have a priority associated with it. If an algorithm other than the scheduled start/stop algorithm is attempting to control an event, it must have a higher priority than the scheduled start/stop algorithm in order to successfully control the event.

5.2 Installation of Scheduled Start/Stop Algorithm

The scheduled start/stop algorithm was installed on the NBS laboratory EMCS. The algorithm was installed in a field processor, integrated into the field processor software. The public domain scheduled start/stop algorithm developed at NBS was described in reference [2] in a general form. An example implementation was written in the high level computer language FORTRAN (FORTRAN IV) as a subroutine and was used as the test algorithm. This subroutine was listed and documented in reference [2].

As originally designed, the scheduled start/stop algorithm as implemented in the field processor software could schedule up to 25 events to occur on a selected day of the week at any time of day with a resolution of one minute. Each event could be scheduled for multiple times as long as the times were greater than one hour apart.

5.3 Corrections to Original Algorithm

No problems were discovered with the original scheduled start/stop algorithm as originally written. Therefore, no corrections were made to the algorithm during the testing.

5.4 Testing of Algorithm

The public domain scheduled start/stop algorithm was tested by allowing an actual air handling unit, which is described in the introduction, to be shut down and started up each day so that the unit was off during the unoccupied period of the building. The NBS laboratory EMCS field processor contained software to start and stop the air handling unit. The air handling unit start/stop software was scheduled for execution using the scheduled start/stop algorithm.

5.4.1 Test Procedure

The scheduled start/stop algorithm was used to start and stop the air handling unit for three separate periods of time. The first test period extended from September 5, 1984 to September 20, 1984. Following this period, tests were performed on the optimum start/stop algorithm and the scheduled start stop algorithm served as a backup to ensure that the air handling unit would start if the optimum start/stop algorithm did not start or stop the unit before a certain times of day. Following optimum start/stop algorithm testing, the scheduled start/stop algorithm was used between December 11, 1984 and January 22, 1985. The third period of testing of the algorithm occurred between February 14, 1985 and March 23, 1985. Altogether there were 92 days of testing with one start and one stop per day. The start was usually scheduled for 06:00 and the stop for 18:00 on all days of the week.

The NBS laboratory EMCS central computer was able to receive an alarm report from the field processor whenever the air handling unit was started or stopped. The alarm reports were stored in a file and displayed on an alarm console along with the exact time of occurrence of the event. The time reports made it possible to verify that the scheduled start and stop did occur at the correct time as scheduled.

5.4.2 Test Results

The evaluation of the performance of the scheduled start/stop algorithm was based on whether or not the algorithm caused the air handling unit to be stopped and started at the correct time. On all 92 days of testing, it was observed that the unit was always started and stopped on time.

5.5 Considerations for Use of Algorithm

The public domain scheduled start/stop algorithm performed as expected. Since it is not a complex algorithm, this is not suprising. No attempt was made to load the algorithm by scheduling the maximum number of events. Also no testing of the algorithm in conjunction with the duty cycling algorithm took place using the actual air handling unit.

6. VERIFICATION OF A DUTY CYCLING ALGORITHM

A duty cycling algorithm is used to reduce the energy consumption of a fan, pump, or other energy consuming device when the load that the device is supplying is smaller than the capacity of the device. By reducing the time that the device is operating, the energy consumption is reduced and hopefully the load is still supplied by intermittent operation of the device. A public domain duty cycling algorithm was developed at NBS and was described in a previous report [2]. This is the algorithm that was tested.

6.1 Operation of Algorithm

The operation of the public domain duty cycling algorithm is described in reference [2]. However, a brief description of the algorithm will be given in this section. The algorithm is designed to cycle several loads on and off over a fixed period called the duty cycle interval. The phase of a load is the time which elapses after the beginning of the duty cycle interval before the load is turned off. The off-period of a load is the time that the load remains off before it is turned on. These duty cycle parameters are stored in a table and are retrieved from the table for each load. The off period for a load is then adjusted by the algorithm as a function of a measured value from the EMCS such as outside air temperature. The algorithm calls a software routine called a load controller to turn each load off after a delay equal to the phase and back on again after a delay equal to the phase plus the off-period.

The load controller is used to coordinate control of loads with other load control algorithms such as demand limiting and scheduled start/stop. This coordination is accomplished through the use of a priority structure for the control of loads. The load control algorithm also provides a way to implement avoidance of loads being on for less than a minimum on-period, off for less than a minimum off-period, or off for more than the duty cycle off-period because the load has been previously turned off by another algorithm. The load controller uses an EMCS dependent routine to actually turn the loads on and off with a certain priority.

The parameters to be supplied for the duty cycling algorithm for each load to be cycled are:

1. phase - time interval between start of duty cycle interval and turning off of the load, in percentage of the duty cycling interval.

2. off-period - amount of time that the load is to be off at design conditions, in percentage of the duty cycling interval.

3. enable adjustment - a logical true if the off-period is to be adjusted as a function of an associated analog variable, or a logical false if the off-period is to remain constant at the design value.

4. design point - a value of an assocated analog variable at which the off-period equals the value described in 2 above.

5. low point - a value of an associated analog variable lower in numerical value than the design point at which the off-period becomes zero.

6. high point - a value of an associated analog variable higher in numerical value than the design point at which the off-period becomes zero.

6.2 Installation of Algorithm

The duty cycling algorithm was installed on the NBS laboratory EMCS. The algorithm was installed in a field processor, integrated into the field processor software. The public domain duty cycling algorithm developed at NBS was described in reference [2] in a general form. An example implementation was written in the high level computer language FORTRAN (FORTRAN IV) as a subroutine and was used as the test algorithm. This subroutine was listed and documented in reference [2].

As originally designed, the duty cycling algorithm example implementation could control up to 16 single digital outputs. Tests were originally run using a test device with lights which could be cycled on and off. The objective of the testing described in this report was to test algorithms on actual HVAC equipment. The only equipment available for duty cycling at NBS was a single air handling unit, and this unit was eventually used for testing. However, cycling of an air handling unit is much more complicated than the cycling of a simple load. Cycling off an air handling unit includes the stopping of at least one and possibly two fans, closing of outside air dampers, and shutting all the valves for the heat exchangers in the unit. To stop the fans in the NBS air handling unit, it was necessary to provide a one second pulse on one digital output, and to start the fans, a one second pulse was required on another digital output. Due to the requirements for control of the NBS air handling unit, it was necessary to make changes to the digital output control software in the field processor. By introducing the concept of a special digital output point which when encountered would cause the air handling unit to be controlled, it was possible to avoid any changes in the duty cycling algorithm load controller. Two additional control modes were added to the air handling unit control software in the field processor. One mode was used to cause the air handling unit to cycle off when the air handling unit control software next executed, and the other mode indicated that the air handling unit was currently cycled off. To allow a scheduled stop of the air handling unit to override duty cycling, appropriate digital output control priority values were selected.

6.3 Corrections to Original Algorithm

One problem appeared in the example FORTRAN implementation of the duty cycling algorithm as listed in reference [2]. This was not an error in the basic algorithm but was a machine dependent problem which resulted in an overflow of an integer variable which contained the length of the off-period in seconds in the FORTRAN used for the programming. Previous testing of the example duty cycling implementation had been performed using artifically short duty cycle intervals of five minutes or less. When a duty cycle interval of 20 minutes was used, the algorithm did not function correctly. The problem was corrected by using a floating point real variable in place of the integer variable originally used for the off-period length.

6.4 Testing of Algorithm

The public domain duty cycling algorithm was tested by cycling an actual air handling unit installed in an operating building. The air handling unit used is described in the introduction section. In order to avoid any potential discomfort to building occupants during the testing, testing was restricted to a Sunday when the building was unoccupied. Only one day of testing was performed due to scheduling constraints.

The duty cycling parameters used for the testing of the algorithm were either 20 minutes or 60 minutes for the duty cycling interval, 33 percent for the design off-period, and 0 percent for the phase. The adjustment of the off-period in response to changes in an associated analog variable was enabled, using the outside air temperature as the associated variable. The design point at which the off-period was a maximum was set at 40 F. The low point and high point were set at 32 F and 60 F, respectively.

6.4.1 Test Procedure

The test procedure consisted of running the duty cycling algorithm for two test periods and recording temperatures in the air handling unit and one of the offices supplied by the air handling unit. The first test period was on March 24, 1985, starting at approximately 15:30 hours and lasting for approximately two hours. For this first test, the duty cycle interval was fixed at 20 minutes.

The second test period began at midnight on March 24, 1985, and lasted approximately seven hours. For this test the duty cycle interval was set at 60 minutes.

6.4.2 Test Results

Figure 6-1 is a plot of the space temperature, supply air temperature, and outdoor air temperature during the first test period. The outdoor air temperature remained constant during the test at approximately 40 F. The supply air setpoint for the unit was 65 F, as indicated by the supply air temperature curve at time 0 on the figure. For each off-period in the duty cycle interval, the fan remained off for approximately 6.5 minutes. While the fan was off, of course, the supply air temperature is not really meaningful. In the air handling unit used for the test, the supply air temperature during the off-period is the temperature of stagnant air in the vicinity of the supply fan. When the fan was turned back on after the off-period, the supply air temperature approached a lower value which was not the supply air setpoint. The setpoint was not reached due to a limitation in the NBS DDC control hardware. Before the duty cycling test, the valve on the heating coil in the air handling unit was open approximately 20 percent. When the unit was cycled off, this valve was closed. The control hardware can overshoot the closed position on the valve. During the 13 minutes that the air handling unit was on, the error in setpoint was insufficient to cause the heating valve to be reopened. In general the DDC control system was designed for continuous rather than intermittent operation. During the test interval, for the NBS building, figure 6-1 shows that the space temperature did not change. The offices in the building were, however, unoccupied.

Figure 6-2 shows the results for the second test period, which are very similar to the results shown in figure 6-1. As with the previous test, the outdoor air temperature remained constant, the supply air setpoint was not reached, and the space temperature was essentially unaffected. A very slight oscillation was noted in the space temperature of less than 0.5 F. Note that a period of nearly three hours was required before the heating valve opened and returned the supply air temperature to the setpoint. Had the test been performed at a different time of the year, or under occupied conditions, it is likely that the control system would have returned the supply air temperature to the setpoint during the duty cycle interval.

In neither of the duty cycling tests, unfortunately, did the outside air temperature change enough to cause a substantial change in the duty cycle off-period. This feature was tested on a building emulator device connected to the NBS laboratory EMCS before the actual testing was performed.

6.5 Considerations for Use of Algorithm

In general, the public domain duty cycling algorithm performed as expected. The air handling unit was cycled on and off at the proper times. Deficiencies in the EMCS DDC control system caused the supply air temperature to be offset from the setpoint. The following sections present any conclusions about parameter selection or general usage reached in addition to the information presented in reference [2].

6.5.1 Parameter Selection

Unfortunately, insufficient experiments were performed to provide guidelines on the selection of duty cycling parameters in addition to those in reference [2]. Additional testing under several conditions of space load, with variations in the parameters, and observations on the results of changes in the parameters would be required to produce definitive guidelines for parameter selection.

6.5.2 Usage Constraints

An issue important to the use of duty cycling of HVAC equipment is whether the cycling causes damage to the equipment. Unfortunately, the test period was insufficient to allow any potential damage to appear. During the test period no signs of damage were observed. The control software to start and stop the air handling unit included safeguards which prevented the fan from being restarted less than one minute after being stopped and prevented the fan from being stopped less than one minute after it had been started.

6.6 Revised Duty Cycling Algorithm

An example implementation of the revised duty cycling algorithm is contained in Appendix B of this report.

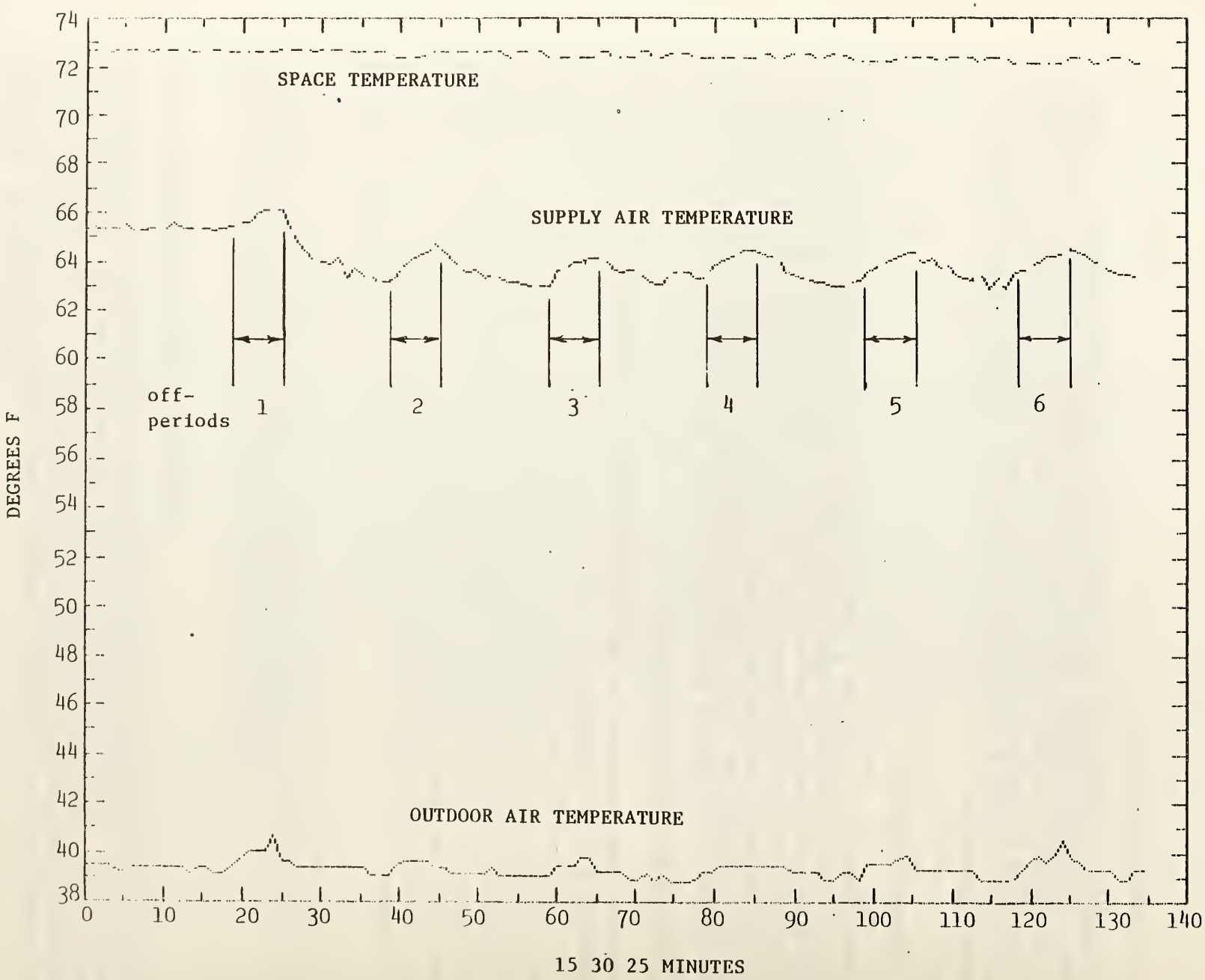

Figure 6-1. Results from duty cycling algorithm test. 20 minute interval.

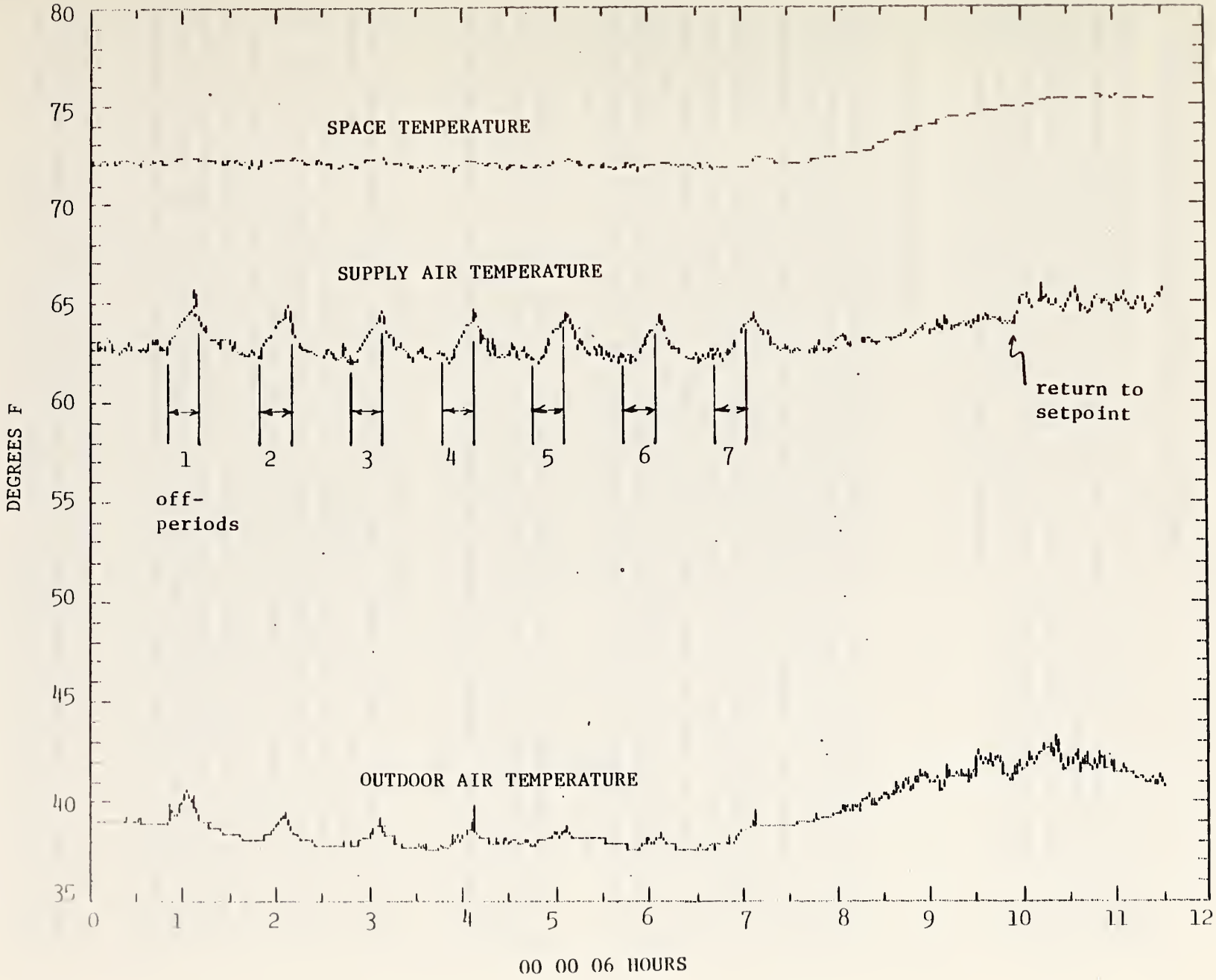

Figure 6-2. Results of testing duty cycling algorithm, 60 minute interval

7. VERIFICATION OF DEMAND LIMITING ALGORITHMS

Demand Limiting Algorithms are used to minimize electrical utility charges associated with electrical demand. The development of public domain algorithms in this area resulted in the production of several different demand limiting algorithms. A detailed description of the algorithms may be found in a previous report [5]. The algorithms developed were instantaeous rate, ideal rate with fixed demand interval, predictive with fixed demand interval, and predictive with sliding window demand interval.

7.1 Operation of Algorithms

The operation of the demand limiting algorithms is well described in reference [5]. However, a brief description of the algorithms will be given in the following paragraphs. All of the algorithms monitor an electrical demand measuring device to determine current electrical demand. The output of the algorithms is a specification of how much electrical power need to be shed or restored at a particular time. Along with the demand limiting algorithms, a load control algorithm was developed which takes this demand limiting output specification and attempts to control electrical loads to shed or restore the appropriate amount of power. By dividing the algorithms into load control and demand parts, it becomes unnecessary for the demand limit algorithms to have any knowledge of the loads that are under control. Also, the same load control algorithm can be used for all the different demand limiting algorithms.

The instantaneous rate demand limiting algorithm is the least complicated of the algorithms. It monitors the instantaneous electrical demand and if the demand is above an upper "high limit," calls for the shedding of demand equal to the difference between the limit and the excessive demand. When the demand drops below a lower "high limit," a call for restoration of demand is made.

The ideal rate demand limiting algorithm requires that the utility or some other source generate a demand interval reset signal which divides time into demand intervals. Often the demand charges are based on the highest total energy used in any demand period as an approximation of real peak demand. The ideal rate algorithm assumes that there is a maximum amount of energy that can be used in a demand interval and that the cumulative energy rises in a linear fashion from zero at the start of the interval to the maximum at the end of the interval. This is the 'ideal rate' energy curve. The measured cumulative energy use is compared to the 'ideal rate' energy curve and if the measured rate of energy increase is higher then demand must be shed. If the measured rate of energy increase falls below the 'ideal rate' then demand may be restored.

The predictive demand limiting algorithms make a prediction of what the cumulative energy use will be at the end of the demand interval, either fixed or sliding window, based on the trend in demand in the early part of the

interval. The predictive algorithm developed uses a simple linear prediction. If the predicted energy exceeds a maximum then demand must be shed.

The load control algorithm which was developed to support the demand limiting algorithms must have information on the electrical loads which are under demand limit control. This information includes the nominal power of the load, the load name, the maximum off time, the minimum off time, the minimum on time, the load activation delay, and the load priority. These parameters are discussed in section 7.5.1. The load control algorithm uses these parameters to select which loads should be shed in order to accomplish a specified demand shedding or restoration goal. The decision is based on the load priority and the power of the load. Loads may be ineligible for shedding if they have been on less than the minimum on time. Loads may be ineligible for restoration if they have been off less than the minimum off time. If loads are at the same priority, the algorithm has features to shed loads at the same priority in a certain order and restore them in the same order. The order for shedding is rotated so that loads at the same priority are not always shed in the same order. One other feature is that the algorithm will automatically restore a load if it has been turned off for an interval greater than the specified maximum off time.

7.2 Installation of Algorithms

Each of the demand limiting algorithms described above was installed on the NBS laboratory EMCS. The algorithms were installed on the central control computer rather than on the field processor. Since the original algorithm was written in FORTRAN 77 and the FID software was written in FORTRAN IV, this was much easier than implementing the algorithm in the field processor. A loss of communications between the central computer and the field processor would have resulted in the demand limiting algorithm being unable to function. However, the placement of the algorithm at this level allows control of demand using all loads available on the system.

All of the other algorithms were tested using actual building spaces and equipment. However, in the test building there were no electrical demand meters available and it would have been difficult to obtain permission to control sufficient electrical loads to test the algorithms. To allow testing of the algorithms on realistic hardware, a special demand emulation device was used. This device consisted of eight light bulbs which could be turned on and off by actuation of relays. In addition, relay contacts were closed when current flowed through the bulbs allowing the status of the loads to be determined. The control relays were connected to eight digital outputs on an NBS lab EMCS field processor multiplexer and the status relays were connected to eight digital inputs on the multiplexer. Thus the EMCS could control the loads and determine their status. The electrical supply to the demand emulation device was routed through a watt-hour meter which had an internal contact closure device. Each change of state of the contact (open to closed or closed to open) indicated that 0.5 watt-hours of energy had been used. This

contact was connected to a special counter input on the EMCS field processor multiplexer. The EMCS was then able to count the contact closures and openings to determine electrial energy use by the demand emulation device. The demand was determined by taking two readings of the meter to get an energy used within the interval between the readings, and dividing the difference in the readings by the time interval. The only disadvantage of using the demand emulator device was that the electrical loads could only be turned on in steps and did not have demands that varied with time as might be experienced in an actual building with electrically powered heating and cooling equipment.

Two special subroutines in FORTRAN 77 had to be written to allow the original algorithms as listed in [5] to control the loads and obtain measured electrical demand. The subroutine to obtain demand requested the current point data base from the field processor and examined the state of the counter input connected to the watt-hour-meter mentioned above for the demand emulation device. The demand was calculated by subtracting the value of the counter at the previous time the subroutine was called from the current counter value and dividing by the time since the subroutine was last called. The disadvantage of this approach is that if power consumption was low and the demand subroutine was called with too high a frequency, the demand values might not be accurate. In the testing, thirty seconds was allowed to elapse between counter readings. In the original demand limiting algorithms, demand was read from a file. This step was replaced with a call to the demand determination subroutine. As an additional benefit, the demand determination subroutine placed values of the digital inputs in a FORTRAN common block variable which was accessible for determining the state of the loads.

The load control subroutine allowed the digital outputs connected to the field processor to be turned on or off by the calling program. The original load control algorithm subprogram used an array of logical variables to represent the states of the digital outputs. The appropriate variable was set to be true or false to control the load. This scheme was replaced with a call to the load control subroutine. The original logical variables were then set depending on the state of the load status input variables determined by the call to the demand determination subroutine.

The demand reset signal for determining the start of the demand interval was obtained by monitoring the state of a digital output in the field processor. This output was assumed to be pulsed on and then off by an external controller (see section 7.4).

One additional change was made to the demand limit algorithms to allow them to operate on the EMCS. The original algorithms contained a program loop which endlessly checked demand and calculated required demand adjustments. A delay was placed in the loop to restrict the execution of the algorithm to once every thirty seconds.

7.3 Corrections to Original Algorithm

Other than the changes required to adapt the demand limiting algorithms to use on a specific real EMCS, no corrections were made to the algorithms. One correction was made, however, to the load control algorithm. The original algorithm assumed that all loads could be restored by a demand limiting algorithm if they were off, even if the load had been turned off by another means and had not been directly shed by the algorithm. The testing of the algorithms on the EMCS required that the loads be externally controlled to simulate changes in demand and therefore this characteristic was unacceptable. To correct the problem, an additional logical variable array was added to the program. There was a logical variable for each load under control. This variable was initially set false. If the load were on and then shed by the demand limit algorithm, the logical variable would be set to true. Only if the variable were true, then, could the demand limiting algorithm attempt to turn on the load to restore electrical demand. The logical variable name used was SHED.

7.4 Testing of the Algorithm

In order to test the demand limiting algorithms using the demand emulator device it was necessary to be able to drive the device to produce a specific demand profile. This could have been done manually by turning the lights in the device on and off. Instead, a special program was written which was designed to run in parallel with the demand limiting algorithms on the central EMCS computer. This special program turned the lights in the demand emulator on and off in a pattern which could be read from data in an input file. This program was also capable of generating the demand interval reset signal at any desired interval.

In order to allow parallel control of the digital outputs connected to the demand emulator device, it was necessary to use the concept of digitial output control priority which is built into the EMCS software. Any output in the field processor can be controlled at a priority level from one to 128 where one is the highest priority. The priority may be unconditionally set, or alternatively, the priority may be requested. If control is requested at a certain priority level and the output is currently controlled at a higher priority, then the control request is rejected. If the load is currently controlled at a priority lower than the requested priority then the control action is taken. For the demand emulator device, the demand emulation program turned on outputs at an unconditional priority of ten and turned them off at an unconditional priority of three. The demand limit load control algorithm requested outputs to be turned off or on with a priority of four. This effectively made the demand limit algorithm unable to turn on loads that the demand emulation program had turned off. Only if the demand limit algorithm had turned off a load could it be turned on by the demand limit algorithm.

7.4.1 Test Procedure

Each of the algorithms listed in section 7.1 were tested individually. The predictive algorithms were actually combined into a single routine which could use either fixed interval or sliding window metering, and was able to switch from fixed interval to sliding window if the demand interval reset signal was lost.

A specific pattern of demand was prepared and used to drive the demand emulator device program for all tests. The demand curve used is shown in figure 7-1 as the solid line. The eight lights in the emulator were turned on in sequence and then turned off in the same sequence. At one point all lights were turned on for a brief period to produce a demand spike. The tests were accelerated to reduce the duration of testing, with the duration of each test being approximately one half hour. The demand limiting algorithms were set to execute periodically with a period of thirty seconds. The demand interval used for all tests was two minutes. Table 7-1 contains the description of the electrical loads used as input to the algorithm for all tests.

table 7-1. description of electrical loads for demand limiting tests.

ID	ITEM	PRIORITY	NOMINAL POWER	DELAY	MINIMUM OFF-TIME (min)	MINIMUM ON-TIME (min)	MAXIMUM OFF-TIME (min)
1	BULB No. 1	1	125.00	0.00	0.00	0.50	10.00
2	BULB No. 2	2	125.00	0.00	0.00	0.50	1.00
3	BULB No. 3	3	125.00	0.00	0.00	0.50	10.00
4	BULB No. 4	2	125.00	0.00	0.00	0.50	10.00
5	BULB No. 5	1	125.00	0.00	0.00	0.50	10.00
6	BULB No. 6	2	125.00	0.00	1.00	0.50	1.50
7	BULB No. 7	1	125.00	0.00	0.00	0.50	10.00
8	BULB No. 8	2	125.00	0.00	0.00	0.50	2.00

7.4.2 Test Results

The maximum demand that the demand emulator device was capable of producing was 1050 watts. In all tests it was assumed that the goal for the demand limiting algorithms was to keep demand below 700 watts. In tests where a demand interval was used, either sliding or fixed, the goal was to keep the integrated energy use during the demand interval below 23.33 watt-hours which was the product of the 700 watt demand and the two minute demand interval.

7.4.2.1 Instantaneous Rate Algorithm

Figure 7-1 shows the results from testing the instantaneous rate demand limiting algorithm as described in section 7.4.1. The solid line represents the theoretical demand that would have been present if no demand limiting had been applied. It shows a series of step functions which would result from each of the eight light bulbs being turned on or off by the demand emulator. The dashed curve represents the electrical demand measured at each execution of the demand limiting algorithm (thirty second intervals). The algorithm was operated with the specification that the maximum demand be lower than 700 watts but higher than 600 watts. The figure shows that the measured demand follows the theoretical demand until the theoretical demand exceeds 600-700 watts and then oscillates around 700 watts. Since the resolution of demand control is only 125 watts (the power consumed by one bulb), these results are acceptable because the amplitude of the oscillation is approximately of this magnitude. A system with a larger number of loads than eight would allow a greater demand control resolution and would allow better control around the maximum demand limit.

7.4.2.2 Ideal Rate Algorithm

Figure 7-2 shows results of testing the ideal rate algorithm as described in section 7.4.1. As in figure 7-1, the solid line represents the theoretical demand if no demand limiting were applied and the dashed curve represents electrical demand measured at thirty second intervals when the demand limiting algorithm is executed. The parameters for the ideal rate algorithm are different than for the instantaneous rate algorithm. The ideal rate algorithm was assumed to operate with a fixed demand interval, and the size of the demand interval, two minutes, was an algorithm parameter. A second parameter was the sampling period or time between successive executions of the algorithm. To establish the ideal rate of energy increase during the demand period, three additional parameters are required. These are the maximum power for the demand period, the offset, and the difference between the minimum and maximum energy curves. The product of the maximum power and the demand interval determines the endpoint of the ideal rate curve on a plot of energy versus time. The endpoint at the start of the demand interval is determined by the offset, which is in terms of energy. The area on an energy vs time plot within which the measured cumulative energy is allowed to fall is determined by the difference parameter. For the actual testing, the maximum power was 700 watts, the offset was 0.25 watt-hours, and the difference between the minimum and maximum energy curves was 0.5 watt-hours. Figure 7-3 is a plot of integrated energy versus time for each of 12 demand intervals during the test period. The dashed line represents the ideal rate curve. The ideal rate line passes between 0.25 watt-hours at the beginning of the demand interval and 23.3 watt- hours (product of 700 watts and 2 minutes) at the end of the demand interval. The solid line represents the measured integrated energy use during the demand intervals which begins at zero for each interval. The algorithm was successful in limiting demand since the solid lines either fall below or

follow the dashed lines in all demand intervals. The demand interval beginning at 12.5 minutes in figure 7-3 is shorter than the other intervals due to an early demand reset signal.

In figure 7-2, which shows the actual demand rather than integrated energy, it may be observed that the demand oscillates with a period of approximately four minutes. The demand interval is equal to half of one period. During a half period, the oscillation generally moves from a maximum to minimum or from a minimum to a maximum, resulting in an average demand (between 600 and 700 watts) when the theoretical demand (i.e., without demand limiting) exceeds these values. The oscillation in demand in figure 7-2 is then more advantageous in reducing fixed interval demand charges than it might at first appear from the large swings in the measured demand.

7.4.2.3 Predictive Algorithm

The predictive algorithm which was tested was able to work with two types of demand intervals, fixed interval and sliding window. The algorithm is capable of switching between the two when the demand interval reset signal is lost. In order to test the algorithm under both types of demand interval, the demand profile shown in figures 7-1 and 7-2 was repeated twice for the predictive algorithm, as shown by the solid line in figure 7-4. The predictive algorithm testing was begun with a demand reset signal appearing every two minutes and the algorithm used fixed interval demand periods. At approximately 30 minutes into the test the demand reset signal was no longer sent and the predictive algorithm switched to the use of a sliding window demand interval for the remainder of the test.

The parameters for the predictive algorithm were the demand interval length, which was two minutes, the maximum and minimum demand levels allowed in a demand period, which were set at 700 and 600 watts to be consistent with tests of the other algorithms, and the sampling period of the algorithm, which was 0.5 minutes. The predictive algorithm attempts to predict the magnitude of integrated energy (integral of power with time) at the end of the demand period and sheds or restores loads to cause the prediction to be less than the maximum and more than the minimum energy parameters (product of maximum and minimum demand parameters and the demand interval). The first half of figure 7-4 shows the result of testing the predictive algorithm with a fixed demand interval. As with the ideal rate algorithm test results, there is an oscillation of demand as the specified demand limit of 700 watts is reached, and the period of oscillation is observed to be approximately four minutes. The magnitude of the oscillation is generally larger than with the ideal rate algorithm. As with the ideal rate, the oscillation results in an average demand during the demand interval which is between 600 and 700 watts whenever the theoretical demand (i.e., without demand limiting) exceeds these values.

Beginning at approximately 30 minutes into figure 7-4, the results of testing the predictive algorithm with a sliding window demand interval are shown. The

demand during the sliding window testing has the same oscillatory characteristics as during the fixed interval testing, but with a greatly increased amplitude. During the period of heaviest demand all of the loads are shed and restored to limit demand. The algorithm apparently produces the drastic oscillations in order to produce smaller integrated energy during the sliding demand periods. It is possible that the oscillatory behavior of the demand would be reduced in a system with a larger number of loads or in a system where the theoretical demand profile was not as rapidly changing as the profile used during this test. Unfortunately, time did not permit testing out these possibilities.

7.4.2.4 Load Control Algorithm

The demand limit algorithm testing also resulted in tests of the load control algorithm. The criteria for judging the load control algorithm are whether loads were reasonably selected for shedding or restoring based on a shed or restore target, and whether the algorithm adhered to parameters for maximum off-time, minimum off-time, and minimum on-time.

From study of the diagnostic output listings from the tests and the load control algorithm parameters in table 7-1, the following observations were made:

1. The maximum off-time limit appears to work properly. During the intantaneous rate test, load 6 was shed and restored after being off for 1.5 minutes which is the specified maximum off-time. Load 8 was subsequently shed to reduce demand restored from load 6.

2. The minimum off-time limit appears to operate correctly. During the instantaneous rate test, load 6 was shed and was not restored until it had been off for one minute, the specified minimum off-time, even though the demand limit algorithm was requesting that demand be restored and only load 6 had been shed.

3. No conclusion is possible about the minimum on-time limit since the limit was set at 0.5 minutes, the sampling period of the algorithm.

4. Some problems were observed during the predictive algorithm testing with a sliding demand window when the algorithm made requests to shed a large amount of demand which exceeded the full load demand of the demand emulator. In this case, the algorithm made requests to shed the same load more than once. This was due to the nature of the modifications made to the load control algorithm where status variables were used to indicate which loads were on and which were off. These were not updated until the next execution of the load control algorithm. Thus, the load control algorithm had no indication that loads shed during the same sampling period had already been shed. The only result, however, was a wasteful increase in the amount of commands sent to the field processor.

7.5 Considerations for Use of Algorithm

In general, the demand limiting algorithms, with the possible exception of the predictive with sliding window demand interval, appeared to work well. The scope of the testing was, however, not sufficient to predict that the algorithms will operate properly under all conditions of loading.

7.5.1 Parameter Selection

This section is intended to document conclusions reached about the assignment of values to algorithm parameters as a supplement to guidelines given in reference [5].

7.5.1.1 Instantaneous Rate Algorithm

The maximum and minimum demand are the parameters for the instantaneous rate algorithm. It was observed that the actual demand exceeded the maximum demand at times. Therefore if it is important that a certain demand figure not be exceeded, a slightly lower maximum value should be used than the actual desired maximum. The maximum demand parameter value seems to serve as a baseline value around which the demand will oscillate when loads are being shed and restored.

7.5.1.2 Ideal Rate Algorithm

The parameters for the ideal rate algorithm and a fixed demand interval are the demand interval, the sampling rate, the maximum power in the interval, the ideal rate curve offset, and the difference between the minimum and maximum energy curves. Due to the limited different selections of parameters used during the testing, no additional conclusions were reached about parameter selection for this algorithm except the recommendation that the sampling period for the algorithm should be less than one quarter of the demand interval.

7.5.1.3 Predictive Algorithm

The parameters for the predictive algorithm used with fixed or sliding demand interval are the demand interval, the sampling rate, and the maximum and minimum power in the interval. Due to the limited different selections of parameters used during the testing, no additional conclusions were reached about parameter selection for this algorithm.

7.5.2 Usage Constraints

The selection of which of the demand limiting algorithms to use depends on the demand metering technique used by the local utility. All of the algorithms tested appeared to work as designed in limiting the demand for the set of loads used for testing. For a small number of loads, all the algorithms produced some oscillation in demand. The predictive algorithm with sliding window demand interval, however, produced very large oscillations, and, because of this, it is recommended that this algorithm not be implemented unless field tests are carried out to determine if the level of demand oscillation which results is within acceptable limits for a particular application.

7.6 Revised Demand Limiting Algorithms

Appendix C contains listings of a specific implementation of the demand limit and load control algorithms as used for testing. Changes to the original programs are delineated in areas separated from the rest of the computer code by dashed lines.

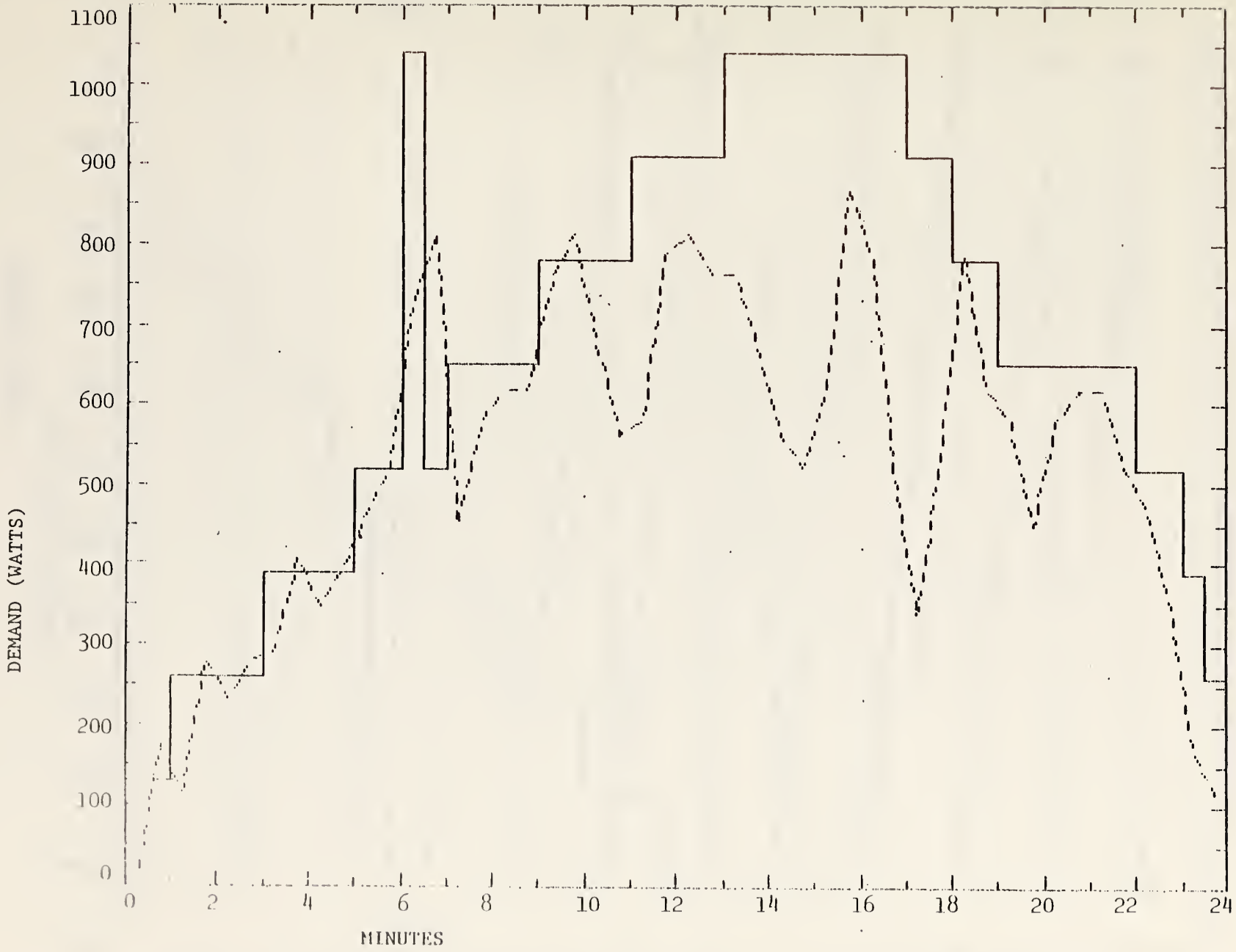

Figure 7-1. Results of testing instantaneous rate demand limiting algorithm

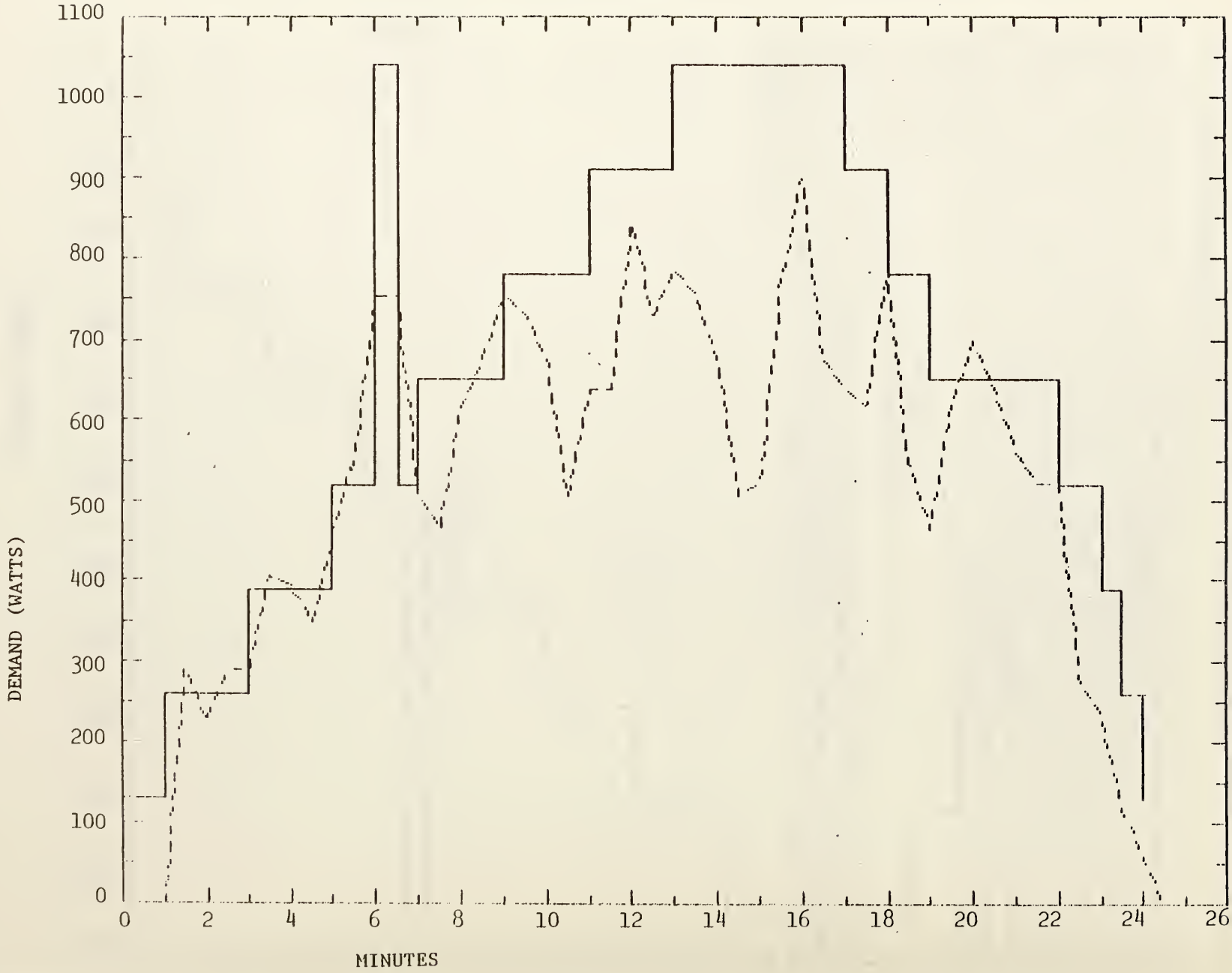

Figure 7-2. Results of testing ideal rate algorithm with fixed demand interval

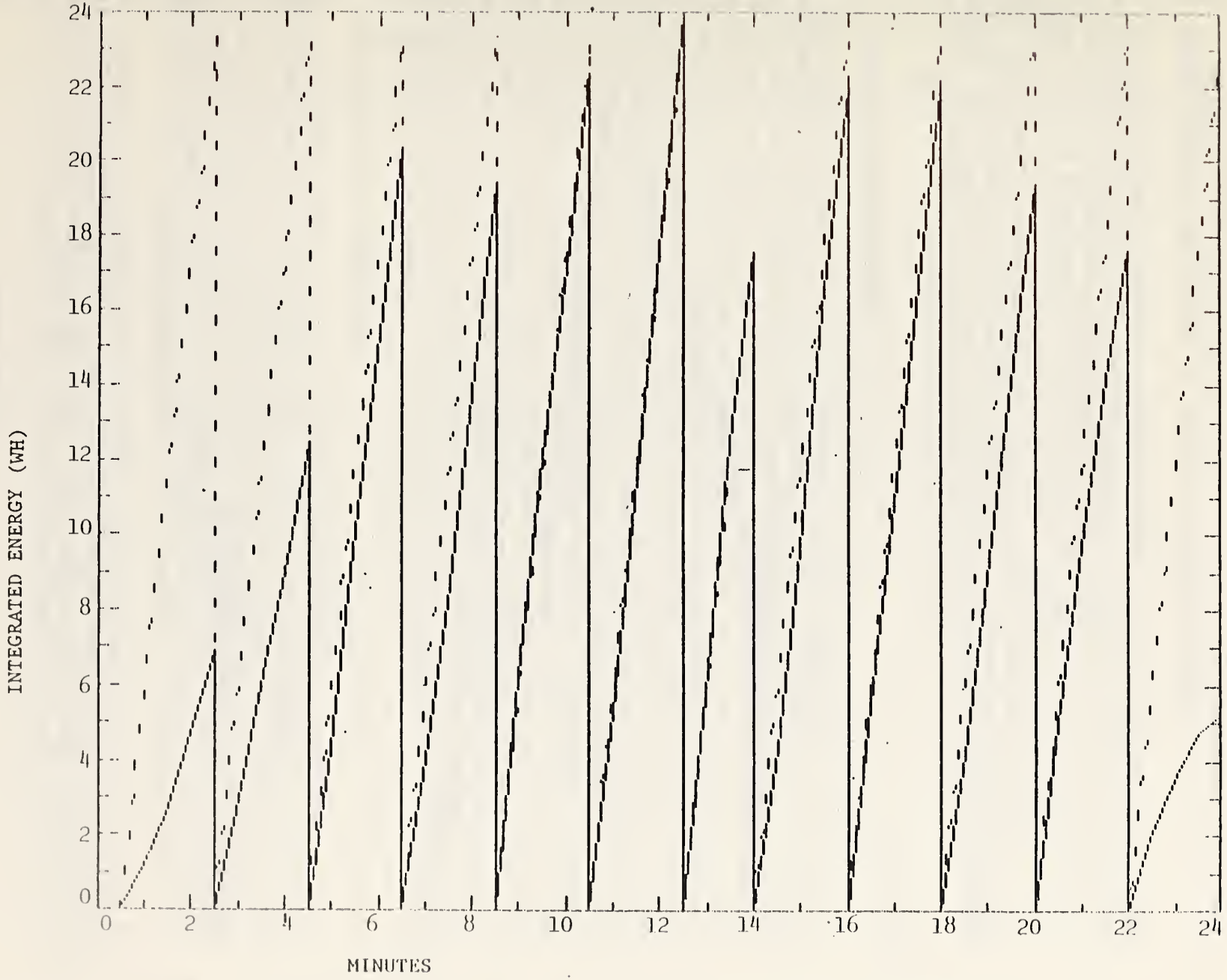

Figure 7-3. Results of testing ideal rate algorithm with fixed demand interval

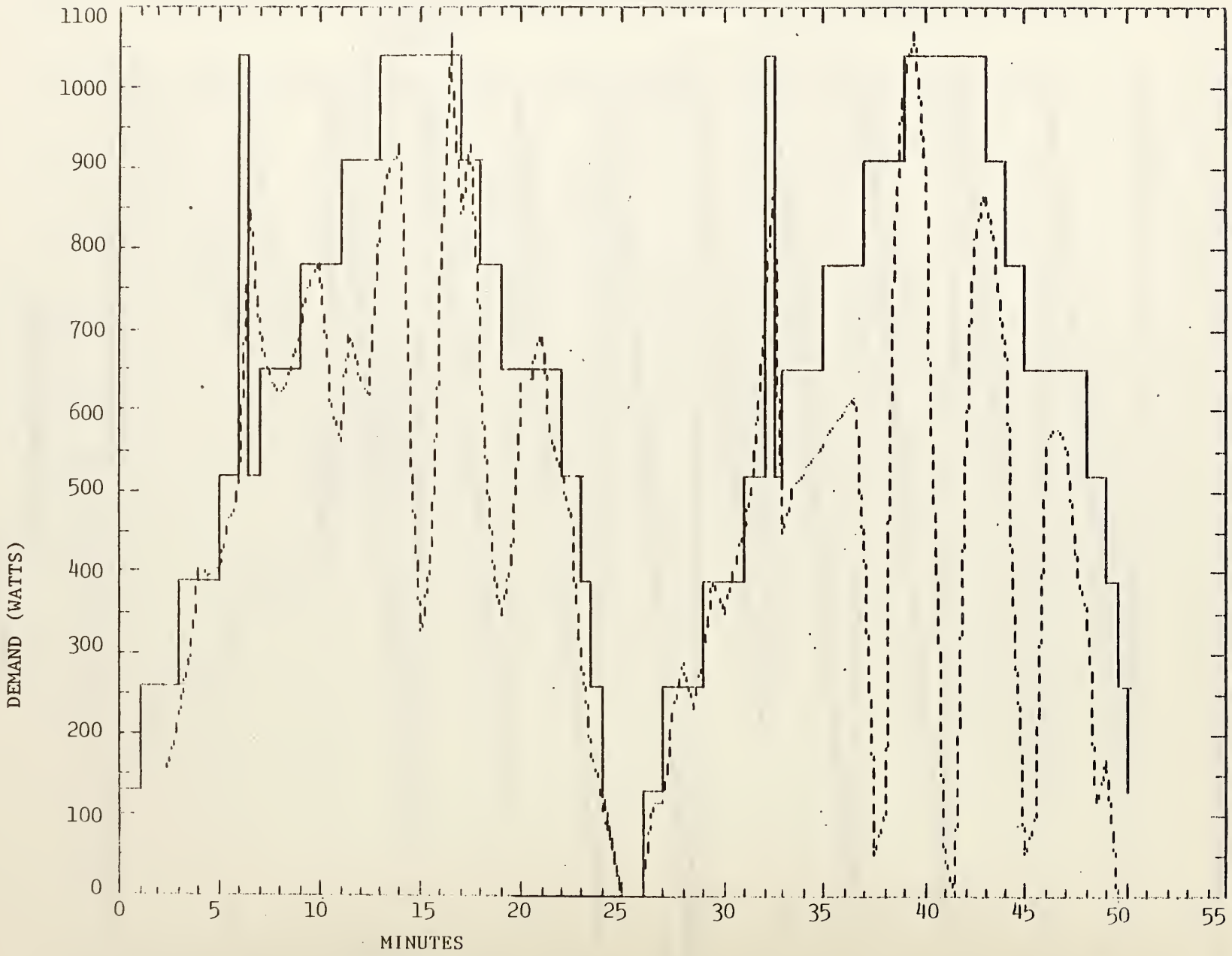

Figure 7-4. Results of testing predictive algorithm with both fixed and sliding window demand intervals

8. VERIFICATION OF AN OUTSIDE AIR SUPPLY AIR RESET ALGORITHM

A reset algorithm installed on a building EMCS is used to change the setpoint of a control loop as a function of a measured variable. When the measured variable is outside air temperature and the setpoint is for the supply air temperature of a building air handling unit, the algorithm is referred to as a supply air reset algorithm. A public domain supply air reset algorithm was developed based on the assumption that the correct setpoint temperature for the air supplied to building zones could be determined as a function of the outside air temperature. A detailed description of the public domain algorithm that was developed is found in a previous report [1].

8.1 Operation of Algorithm

Operation of the public domain outside air reset algorithm is described in reference [1]. However, a brief description of the algorithm will be given in this section. The algorithm is designed to change the values of one or more setpoints according to a predetermined schedule table of setpoints as a function of measured outside air temperatures. The table contains three values of setpoint for three values of outside air temperature. If the outside air temperature is below the first table entry of air temperature, then the setpoint is constrained to the value of the setpoint for the first table entry. If the outside air temperature is above the outside air temperature for the third table entry, then the setpoint is constrained to the value of the setpoint for the third table entry. If the outside air temperature is between the outside air temperatures for the first and second table entries then the setpoint is interpolated between the setpoints for the first and second table entries. If the outside air temperature is between the outside air temperatures for the second and third table entries then the setpoint is interpolated between the setpoints for these two table entries. If the setpoint is plotted as a function of outside air temperature, the result is four straight line segments.

8.2 Installation of Algorithm

The outside air supply air reset algorithm was installed on the NBS laboratory EMCS. The algorithm was installed in a field processor and integrated into the field processor software. The public domain outside air supply air reset algorithm developed at NBS was described in reference [1] in a general form. An example implementation was written in the high level computer language FORTRAN (actually FORTRAN IV) as a subroutine and was used as the test algorithm. This subroutine was listed and documented in this reference.

The setpoints controlled by the reset algorithm were used by the field processor air handling unit control software in DDC control of an air handling unit. The valves and dampers of the air handling unit were adjusted to maintain the air leaving the unit at the desired supply air setpoints.

8.3 Corrections to Original Algorithm

No problems were discovered with the originial outside air supply air reset algorithm as originally written. Therefore no corrections were made to the algorithm during the testing.

8.4 Testing of Algorithm

The public domain supply air reset algorithm was tested by controlling the supply air setpoint temperature of an actual air handling unit installed in an operating building as a function of the measured outside air temperature. The air handling unit that was used is described in the introduction. The algorithm was allowed to control the setpoint continuously for a period of approximately two months using several reset schedules. Measurements were made of the outside air temperature, the supply air temperature, and the air temperature and activity of the reheat coil in one office to evaluate the operation of the algorithm.

8.4.1 Test Procedure

Continuous measured data from a one month interval where the supply air reset algorithm used the same reset schedule every day were selected to evaluate the algorithm. This period was from February 24 through March 23, 1985. The reset schedule used during the test period contained the values given in table 8-1.

During the test period, a scheduled start/stop algorithm was also stopping the air handling unit at 18:00 (6:00 p.m.) hours every day and starting the unit up at 4:00. When the air handling unit was stopped, the supply air temperature was not meaningful because no air was passing through the air handling unit. Therefore only data collected while the air handling unit was operating were used for algorithm test evaluation.

table 8-1. reset schedule used to test outside air supply air reset algorithm

outside air temperature (F)	supply air temperature (F)
40	66
72	60
90	55

8.4.2 Test Results

Since the outside air supply air reset algorithm adjusts the supply air temperature as a function of the outside air temperature, it is reasonable to assume that a plot of measured supply air temperature as a function of measured outside air temperature would be useful in determining if the algorithm is operating as expected. Unfortunately, such a plot will also include the effects of the local control system, in this case a direct digital control system controlling a chilled water valve, an outside air damper, and a steam heating coil.

Figure 8-1 is a plot of measured supply air temperature versus measured outside air temperature for the test period. Data were taken approximately every five minutes, but data for each day between 18:00 and 4:00 on the next day were omitted since the air handling unit was not operating during these time periods. The solid line drawn on the plot represents the outside air reset schedule as given in table 8-1. Theoretically, all points on the plot should fall along this line.

The bulk of the points in figure 8-1 do follow a wide pattern which definitely falls along the schedule line. However, there are three distinct discrepancies, the first of which is that a large number of points lie above the schedule in two patterns. A second discrepancy is the group of points which lie below the reset schedule, and the third discrepancy is the large spread in the points which do follow the schedule.

Unfortunately, as mentioned above, the results in figure 8-1 describe the performance of the DDC control system as well as the reset algorithm performance. Based on observations of the DDC system, it is known that two problems exist with this system. The first is that in order to minimize oscillations, the control system is tuned in such a way that it fairly sluggish to changes in load or setpoint. The second problem is that when sequencing occurs between the fully open or closed position of one valve and the fully closed or open position of the next valve in the sequence, there can be a sizable delay before the next valve responds. The size of this delay depends on the past history of the valve control actions. These valves are controlled by control signals that cause a change in valve position rather than a change in absolute position. A shut down of the air handling unit, such as occurred each day, compounds the problem, because all valves are driven to the closed position and the valve actuator system (in this case, a motorized pressure regulator) can physically overshoot. If the overshoot occurs, a number of control actions will be required to return the valve to the point where it is just starting to open.

In figure 8-1, the discrepancies where the supply air temperature is greater than the scheduled valves are caused by sluggish sequencing when the heating load on the air handler is decreasing. On days when the outside air temperature was below 45 F prior to air handling unit start-up and rose during

the day, the air handling unit usually began to sequence from the steam valve to the outside air damper when the outside air temperature reached 45 F. However, there was usually a delay after the steam valve closed and before the outside air dampers opened. This delay allowed the supply air temperature to temporarily rise, sometimes as high as 70 F. This situation occurred often during the test period and accounts for most of the points in figure 8-1 leading upward from the main group starting at an outside air temperature of 45 F.

On warmer days, the air handling unit would make a transition from outside air dampers to the chilled water coil and the delay in sequencing resulted in the smaller number of points leading upward from the main group around an outside air temperature of 60 F.

Occasionally, there was a condition where there was a delay in the opening of the valve for the steam coil under cold conditions and this allowed the supply air temperature to drop below the setpoint temporarily. This caused the grouping of points in figure 8-1 below the schedule line between outside air temperatures of 35 and 40 F.

The main group of points in figure 8-1 does follow the schedule line with considerable spread. The spread is probably caused by the sluggish DDC control when faced with changes in setpoint or loading. The nature of the control could cause the actual supply air temperature to lag behind the setpoint temperature by as much as two degrees, which is the width of the spread on the points clustered around the reset schedule line.

Taking into account the behavior of the DDC controller, it appears that the data in figure 8-1 indicate that the public domain outside air supply air reset algorithm did perform as expected. Unfortunately, the algorithm was not run during other test periods having a larger range of outside air temperatures. Only a portion of the reset schedule was used. However in tests using a building emulator device connected to the NBS laboratory to simulate a building under a wide range of outside air temperatures, the supply air temperature predicted by the emulator followed the specified reset schedule.

8.5 Considerations for Use of Algorithm

In general, the public domain outside air supply air reset algorithm performed as expected. Unfortunately, the HVAC system connected to the NBS laboratory EMCS was not able to fully benefit from energy savings possible from use of the algorithm because of poor local control of the building offices by terminal reheat units. This problem was encountered in testing of the demand supply air reset algorithm and is discussed in section 9.4.

8.5.1 Parameter Selection

The parameters for the outside air reset algorithm determine the reset schedule to be used by the algorithm. The selection of the proper schedule is important since the control of the supply air temperature is an open loop type of control. If the supply air temperature selected by the algorithm is improper for the load conditions in the building, the algorithm will not correct the setpoint. Only the entry of a revised reset schedule will correct the problem. It was difficult to develop a schedule for the building used for testing of the algorithm since the local zone reheat controllers did not change the amount of reheat as a result of changes in supply air temperature and the proper supply air temperature for various outside air temperature was difficult to determine (see section 9.4).

The air handling unit used for testing has traditionally been controlled at a constant setpoint of 65 F in the winter and 60 F in the summer. The use of the outside air reset algorithm would at least allow the change in setpoint from summer to winter and winter to summer to be automated. This could be accomplished by setting the three outside air temperatures to a small range of outside air temperatures corresponding to summer-winter transition, such as 65, 68, and 70 F. The corresponding setpoints in the reset schedule table might then be 65, 62, and 60 F. The result would be a constant setpoint of 65 F in the winter (outside temperature below 65 F), and a constant setpoint of 60 F in the summer (outside air temperature above 70 F).

8.5.2 Usage Constraints

The use of an outside air supply air reset algorithm is only justified if the HVAC system will use less energy with the algorithm than without it. If the local zone heating and cooling equipment is not working properly or is not properly controlled, the benefits from the reset algorithm will be greatly reduced. If the local equipment is operating properly, the use of the algorithm should save both cooling energy and local zone heating (such as reheat energy) as the supply air temperature will be more closely matched to the zone heating and cooling loads.

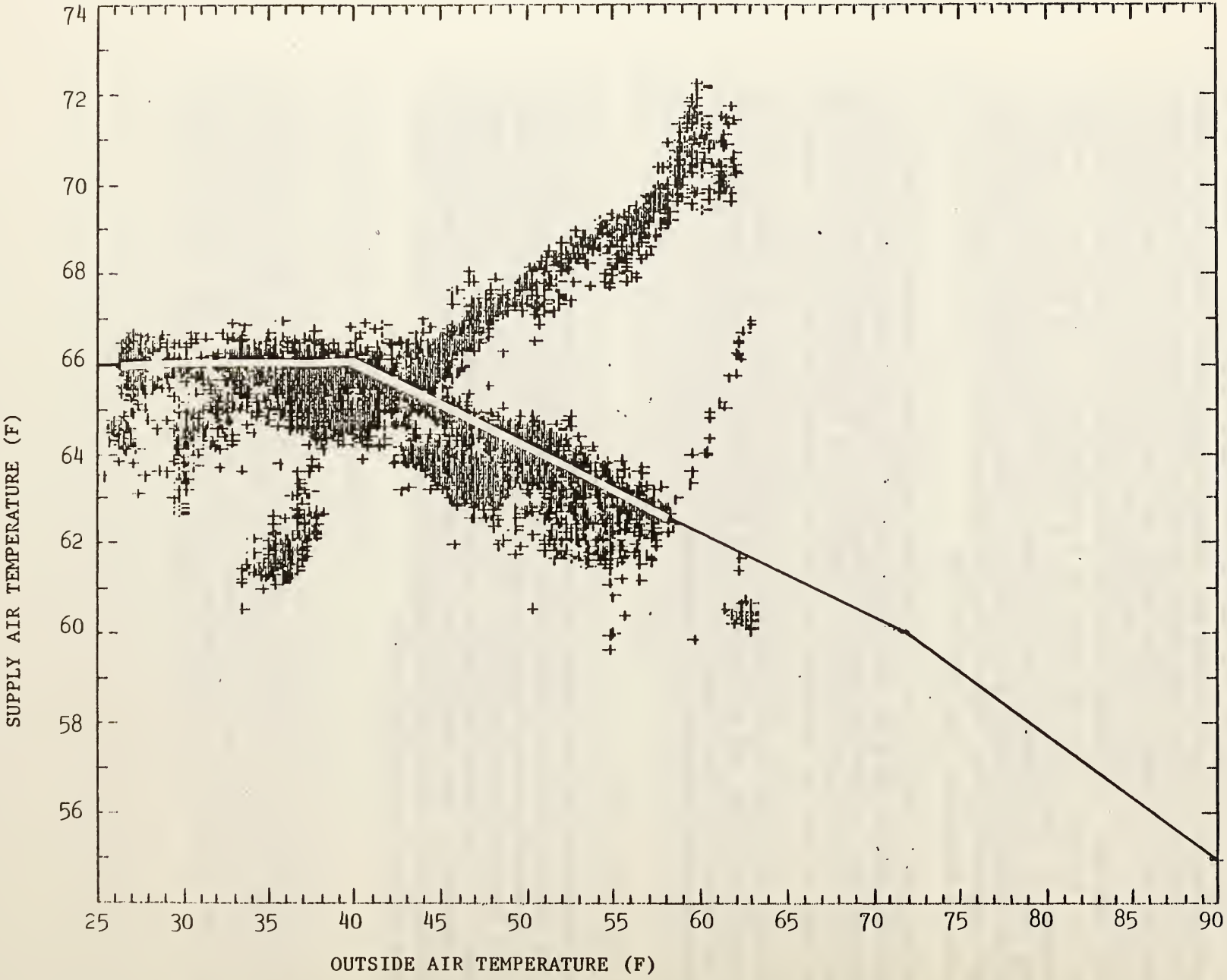

Figure 8-1. Results of testing outside air reset algorithm (2-24 to 3-23-85)

9. VERIFICATION OF A DEMAND SUPPLY AIR RESET ALGORITHM

A reset algorithm installed on a building EMCS is used to change the setpoint of a control loop as a function of a measured variable. When the setpoint is for the supply air temperature of a building air handling unit providing air to one or more building zones and the measured variable is the heating or cooling requirements of the zones, the algorithm is usually referred to as a demand supply air reset algorithm. A public domain demand supply air reset algorithm was developed based on the assumption that the zone heating or cooling demand can be determined by measurements and that the best setpoint temperature of the supply air to building zones can be determined as a function of the zone demand. A detailed description of the public domain algorithm developed can be found in a previous report [1].

9.1 Operation of the Algorithm

Operation of the public domain demand reset algorithm is described in reference [1]. However, a brief description of the algorithm will be given in this section. The algorithm is designed to change the values of one or more setpoints as a function of the measured zone demand. Readings are taken from instrumentation in one or more zones and summed or averaged over a time period which is selectable as an algorithm parameter. The algorithm does not specify the nature of the data taken from the zone, since this data will depend on the zone characteristics and local heating and cooling equipment. After enough samples have been taken, the algorithm will calculate, for each of the selected zones, a value referred to here as the "demand", which represents the percentage of the maximum heating or cooling capacity for the local equipment used over the past sampling period in each zone. Depending on the HVAC system, one of the zones is then selected as having the zone demand which will be used to calculate a new supply air setpoint. For example, with a constant volume air handling unit and terminal reheat units in the zones, the zone with the smallest amount of reheat will be used as the representative zone. The calculation of the new setpoint is made using equations which are dependent on the HVAC system type. Once the new setpoint had been calculated and reset, the sampling of the zone instrumentation begins again for another sampling period.

9.2 Installation of the Algorithm

The demand supply air reset algorithm was installed on the NBS laboratory EMCS. The algorithm was installed in a field processor and integrated into the field processor software. The algorithm, which was developed at NBS, is described in reference [1]. An example implementation was written in the high level computer language FORTRAN (actually FORTRAN IV) as a subroutine. This program was listed and documented in reference [1].

The setpoints controlled by the reset algorithm were used by the field processor air handling unit control software in DDC control of an air handling

unit. The valves and dampers of the air handling unit were adjusted to maintain the air leaving the air handler at the supply air setpoints.

The specific HVAC system type used for the testing of the demand supply air reset algorithm was a constant volume air handling unit with terminal reheat units in the zones. The reheat coils in the zones were theoretically modulating hot water coils controlled by a local thermostat with the thermostat sensor exposed to the zone air. The thermostat and hot water valve were pneumatic devices, without any possible adjustments other than an uncalibrated setpoint selection knob. It was observed, however, that in actual practice the hot water coils were either on or off, even though the control was theoretically modulating. This was attributed to an overly large proportional gain built into the local thermostat. The local instrumentation selected to indicate reheat coil activity was a pressure switch which closed a contact at a certain pressure. The switch activation pressure was adjusted to correspond to the midpoint of the pressure range for the reheat valve. Thus an open valve would produce a closed contact and a closed valve would result in an open contact. This contact was connected as a digital input on the NBS laboratory EMCS. Due to limitations in time and the difficulty of instrumenting all of the 39 zones supplied by the air handling unit, it was decided that instrumenting a single zone would be adequate to test the algorithm for basic functionality.

The specific details of the algorithm operation followed the general approach outlined in section 9.1. A data point was taken from the reheat coil instrumentation described above every fifteen seconds. The control period for reset was set at 1200 seconds, resulting in a total of 80 data points for each control period. If the reheat coil were on constantly, each time the reheat coil sensor was sampled, the contact would be closed, giving a total of 80 contact closures. If the reheat coil operated 50 percent of the time, the sensor would have a closed contact half of the time or 40 contact closures would be counted. By dividing the total amount of contact closures during the control period by the maximum (80), the percentage of the maximum reheat coil usage was determined for each control period. The sampling time and the control period were parameters for the algorithm.

Five other parameters were used by the demand supply air reset algorithm to calculate a supply air setpoint from the percentage of maximum reheat coil usage. These are specified minimum reheat coil usage in percent, the specified maximum reheat coil usage in percent, the reset gain, the minimum supply air setpoint, and the maximum supply air setpoint. The goal of the algorithm is to adjust the supply air temperature so that the observed percent reheat coil usage remains between the specified minimum and maximum reheat values. This is accomplished with a simple proportional control loop with the reheat coil percent of use as the measured or process variable and the change in the air handler's supply air setpoint, relative to the current setpoint as the output variable. The reset gain parameter is multiplied by an error term to yield the correction to the current supply air setpoint. This error term is defined as the current reheat value minus the specified maximum reheat value if the

current reheat value is above the maximum or is defined as the current value minus the specified minimum reheat value if the current value is below the minimum. Once the correction has been determined, it is added to the current setpoint to produce the new setpoint. The new supply air setpoint is compared to the maximum and minimum supply air setpoint parameters and constrained to be within these limits. If the current reheat value is between the minimum and maximum specified values, no correction to the supply air setpoint is made (this is equivalent to assuming that the error term is zero).

The implementation example of the demand supply air reset algorithm listed in reference [1] was not written in a completely general form. The number of zones and the identification of the digital inputs to use for the reheat coil sensors were not designated as algorithm parameters but were written directly into the computer program. These two values had to be changed in order to test the algorithm.

9.3 Corrections to Original Algorithm

One correction to the original public domain demand supply air reset algorithm as listed in reference [1] was made before testing of the algorithm. The original algorithm mistakenly determined the correction to the current supply air setpoint by multiplying the reset gain by the specified minimum percentage of reheat capacity rather than by the error between the setpoint and the minimum. This calculation was changed to use the error instead.

9.4 Testing of Algorithm

The public domain supply air reset algorithm was tested by controlling the supply air setpoint temperature of an actual air handling unit installed in an operating building as a function of the measured reheat demand in one zone supplied by the air handling unit. The air handling unit that was used is described in the introduction. The algorithm was allowed to control the setpoint continuously for a period of approximately two weeks using several different sets of demand reset parameters. Measurements were made of the outside air temperature, the supply air temperature, and the air temperature in one office to evaluate the operation of the algorithm.

9.4.1 Test Procedure

It was discovered, while testing the demand supply air reset algorithm, that the local reheat coil control in the zone which was being monitored did not operate as expected. Theoretically, the amount of time that the reheat coil was operating should vary as a function of three variables. First, changing the reheat coil thermostat setpoint upward should cause the coil to operate more of the time and changing the setpoint downward should cause it to operate less. Secondly, any changes in the zone load due to changes in transmission

loads or internal loads should cause the reheat time to go down with decreased heating loads or increased cooling loads, or it should cause the reheat time to go up with increased heating loads or decreased cooling loads. Thirdly, an increase in the supply air temperature should cause the zone to move towards higher zone air temperatures and therefore lower the amount of reheat. Similarly a decrease in supply air temperature should cause the zone to move towards a lower zone air temperature and should raise the amount of reheat.

By varying the supply air temperature and thermostat setpoint and recording the activity of the reheat coil, it was discovered that the reheat coil essential is activated in pulses of approximately one to two minutes duration. The duration of the pulses seems to vary slightly but the variation in pulse width could not be correlated with any other variables. It was found that changes in the supply air temperature, while having an effect on the room temperature, as would be expected, had no significant effect on the pulse width or frequency of the reheat coil activity. Changes in the thermostat setting of the reheat control did have an effect on the reheat coil activity. If the thermostat were moved toward a setting of higher room temperature, the frequency of the reheat coil pulses was found to increase. If the setting were moved toward a lower room temperature the pulse frequency decreased. Effectively the reheat coil in this room injected a constant amount of energy into the zone, regardless of supply air temperature or load conditions if the thermostat setting was kept constant. Increasing the thermostat setting, for example, simply increased the rate of energy injection. The reason for this behavior is assumed to be due in part to a large proportional gain in the reheat controller, which causes reheat valve position to oscillate between the open and closed positions.

Due to the problems with the zone reheat coil, it was not possible to actually minimize reheat by changing the supply air temperature. Therefore the test procedure consisted of operating the system with the demand supply air reset algorithm operating and changing algorithm parameters and observing the results. Since the behavior of the reheat coil was predictable (constant output), it is to be expected that, if the specified reheat demand maximum and minimum values were not properly set to bracket the reheat coil's measured constant demand, then the algorithm would drive the setpoint to the high or low setpoint limit at a rate which would depend on the control parameters used.

9.4.2 Test Results

The results of testing the public domain demand supply air reset algorithm with a zone containing an improperly working reheat coil are presented in table 9-1. The table shows the parameters that were used and the starting dates and times that the parameters were changed. In all cases the supply air setpoint was initially placed at 60 F. The algorithm immediately began to cause the supply air setpoint to ramp upward or downward at a constant rate until either the high or low setpoint limit was reached and then caused the

setpoint to remain at the limit. The only changes in this behavior were whether the ramp was up or down and the time required for the setpoint to make a transition from 60 F to the upper or lower limits. This duration of the ramp is noted in the last column of table 9-1.

In the second row of table 9-1, when the reheat target was set at 20 to 25 percent, this was obviously higher than the measured reheat coil usage as measured in percentage of maximum capacity. The algorithm attempted to raise the amount of reheat to reach a point between 20 to 25 percent by lowering the supply air temperature. It took four hours to reach the limit of 55 F. When the reheat target was lowered to 15 to 20 percent in the third row of table 9-1, the algorithm again lowered the setpoint but since the error in reheat demand was smaller than for the previous case, it took approximately twice as long to reach the lower limit. Additional lowering of the reheat demand target caused the duration of the ramp to increase to 23 hours for a reheat demand target between 12 and 17 percent. Lowering the target still further to the 10 to 15 percent range caused the target range to lie below the measured reheat coil demand. The algorithm, in order to lower the amount of reheat, moves the setpoint toward the upper setpoint limit as indicated in row 1 of table 9-1.

table 9-1. results of testing public domain demand supply air reset algorithm

control period = 1200 seconds, reset gain = 0.05 F/percent of capacity

starting date	time	% reheat capacity min	max	ramp destination (F)	duration of ramp (hrs)
5-3	16:30	10	15	66	8
5-6	10:35	20	25	55	4
5-8	12:15	15	20	55	8
5-13	11:37	13	18	55	15
5-15	13:30	12	17	55	23

9.5 Considerations for Use of Algorithm

In general, the public domain demand supply air reset algorithm performed as expected with the HVAC system that was controlled. Unfortunately, it was not possible to determine if the algorithm would have provided a direct benefit in reducing the energy used by the air handling unit due to the poor performance of the local zone reheat system. The following section discusses any conclusions reached on selection of algorithm parameters or usage of the algorithm based on the testing of the demand supply air reset algorithm. It should be kept in mind that the conclusions are based on the specific implementation involving the control of the supply air temperature for a constant volume air handling unit with terminal reheat.

9.5.1 Parameter Selection

The parameters used for the demand supply air reset algorithm either control the desired local controller's activity, the range of allowed supply air setpoints, or the determination of a new supply setpoint from the local controller's measured activity.

The parameters determining the range of supply air setpoints are usually dependent on the specific building and HVAC system. During the testing, the maximum and minimum setpoints used were based on the setpoints for winter and summer operation traditionally used for the air handling unit being controlled. The maximum and minimum values selected were expanded slightly beyond the constant setpoints to give the algorithm an adequate working range to minimize reheat activity.

The parameters control period and sampling interval are used to make a determination of the local zone equipment activity. Only one set of values for these parameters was used during the algorithm testing. The control period was chosen as the interval over which it was desired that the setpoint be adjusted. The sampling interval was chosen to be fairly small. In the test case, a digital status point was being monitored. In order to capture the true behavior of the zone equipment, the sampling time had to be at least twice the maximum frequency of the zone equipment activity. In this case the reheat coils were observed to pulse with an approximate frequency of 0.5 cycles per minute. The sampling interval then should have been less than one minute. Dividing the control period by the sampling interval produces the number of samples taken in the control period. If this number is small, it limits the resolution of the zone equipment activity. If the number of samples is 20, for example, the maximum resolution is 5 percent of the maximum zone equipment capacity.

The specified maximum and minimum reheat values and the reset gain are specific to a given application involving a reheat system, but should have counterparts in most implementations. The reset gain is used to convert the deviation in zone equipment activity from the specified maximum and minimum values into a change in the supply air setpoint. Only one value was used during the testing. Since the control of the setpoint does not have to be highly dynamic, the gain was chosen to be small in order to produce a gradual, non-oscillatory response to changes in the reheat coil activity. The behavior of the algorithm in controlling the supply air setpoint was very smooth, which would be expected since the local equipment activity was essentially constant. The specified maximum and minimum reheat values determine the desired value for zone equipment activity. During the testing, unrealistic values were used to verify that the algorithm was operating as expected. In control of a system where the local equipment control was working properly, these parameters would be chosen to be close to zero. The specified minimum reheat value should, however, be greater than zero by an amount that is greater than the possible

resolution of local equipment activity. The latter, in turn, is dependent upon the sampling interval and sampling period. The closeness of the specified maximum and minimum reheat values to each other depends on the behavior of the local control equipment. Local equipment activity readings which indicate that the local equipment control is well behaved and the equipment functions as expected would imply that the specified maximum and minimum activity values should be chosen to be close together.

9.5.2 Usage Constraints

The use of a demand supply air reset algorithm is only justified if the HVAC system will use less energy with the algorithm than without it. If the local zone heating and cooling equipment is not working properly or is not properly controlled, the use of the reset algorithm will provide little benefit. If the local equipment is operating properly, the use of the algorithm should save local zone heating and cooling energy (such as reheat energy) as the supply air temperature will be more closely matched to the zone heating and cooling loads. The use of the demand supply air reset requires adequate instrumentation of the zones to be of greatest benefit. The testing of the algorithm used sensors in only a single zone. This would be inadequate in any building where the loads in the zones supplied by the same air handling equipment are dissimilar. For some types of local zone equipment the instrumentation of the zone to determine local equipment activity may not be practical.

9.6 Revised Demand Supply Air Reset Algorithm

Because of errors found in the original demand supply reset algorithm contained in reference [1], a revised listing of the algorithm, as implemented in this testing program, is provided in Appendix D.

10. SUMMARY

This report has described the testing and verification of eight supervisory control algorithms, which were previously developed for Energy Management and Control Systems (EMCS) by the National Bureau of Standards. The algorithms tested covered dry bulb and enthalpy economizer cycles, optimum and scheduled start/stop, duty cycling, demand limiting, outside air supply air reset, and demand supply air reset.

The algorithms, which were described previously in reports [1,2,3,4,5], were evaluated using an EMCS system developed in the laboratory at NBS. For each algorithm, the process of installing the algorithm on the NBS laboratory system was discussed and a description given of the tests that were carried out to evaluate its performance. The results from these experimental studies were presented, along with any additional considerations for use of the algorithms that were developed as a result of the testing program.

Four of the algorithms - dry bulb and enthalpy economizer cycles, scheduled start/stop and outside air supply air reset, performed as they were designed to without any significant changes. The remaining algorithms all required some code modifications in order to perform properly. For these algorithms, the changes that were made were discussed and a listing of the revised program code, as specifically implemented on the NBS laboratory EMCS system, was provided in appendixes.

11. REFERENCES

[1] May, W. B., "Control Algorithms for Building Management and Control Systems - Hot Deck/Cold Deck/Supply Air Reset, Day/Night Setback, Ventilaton purging, and Hot and Chilled Water Reset," Nat Bur. Stand. (U.S.) NBSIR 84-2846, March 1984.

[2] May, W. B., "Time of Day Control and Duty Cycling Algorithms for Building Management and Control Systems," Nat. Bur. Stand. (U.S.) NBSIR 83-2713, June 1983.

[3] Park, C., "An Optimum Start/Stop Control Algorithm for Heating and Cooling Systems in Buildings," Nat. Bur. Stand. (U.S.) NBSIR 83-2720, May 1983.

[4] Park, C., Kelly, G. E., and Kao, J. Y., "Economizer Algorithms for Energy Management and Control Systems," Nat. Bur. Stand. (U.S.) NBSIR 84-2832, February 1984.

[5] Park, C., "Demand Limiting Algorithms for Energy Management and Control Systems," Nat. Bur. Stand. (U.S.) NBSIR 84-2826, February 1984.

APPENDIX A. SAMPLE IMPLEMENTATION OF OPTIMAL START/STOP ALGORITHM

```
C ******************************************************************
C
C     OPTSS : OPTIMUM START AND STOP CONTROL ALGORITHM
C
C     MAR. 3, 1983 C.P.
C     MODIFIED JULY 27, 1984 - NPTHR passed through COMMON from main
C      (W.B. MAY)              program.
C                            - Check for zero time constants in YONNEW
C     MODIFIED AUG. 2, 1984  - MAXMIN subroutine arguments now include
C      (W.B. MAY)              range of search.
C                            - Determination of max,min outdoor temps.
C                              modified to use different search range
C                              on first pass of algorithm.
C                            - All comparison criteria (epsilons)placed
C                              in common block for external assignment
C     MODIFIED AUG. 13 1984  - Ability to respond to start/stop failure
C      (W.B. MAY)              alarms is added. Flags are set to bypass
C                              parameter updating and response waiting.
C     MODIFIED AUG. 24 1984  - Added warm start capability. After the
C      (W.B. MAY)              "morning" calculations are done,
C                              parameters are saved in a file. If the
C                              program is restarted, it attempts to
C                              read parameters from this file. If not
C                              successful, the normal cold start is
C                              used.
C     MODIFIED AUG. 29 1984  - Added additional statement to improve
C      (W.B. MAY)              above change and prevent multiple reads
C                              of the save file. Also added if clause to
C                              prevent the on-off controller from
C                              running in the daytime.
C     MODIFIED AUG. 30, 1984 - Added additional variables to be saved in
C      (W.B. MAY)              save file to ensure correct operation.
C
C     MODIFIED SEP. 27, 1984 - Additional precautions added against division
C      (W.B. MAY)              by zero, algorithm failure at low loads.
C
C     MODIFIED OCT. 01, 1984 - Added limit to earliest stop time AND ...
C      (W.B. MAY)              added ability to switch between heating and
C                              cooling mode. If the algorithm cannot find the
C                              root of the start-up and shut-down curves,
C                              the mode is switched, which changes the YSS and
C                              flips the start-up curve about the time axis
C                              at the setpoint temperature, allowing a root
C                              to be found.
C-----------------------------------------------------------------------
C
C  DAYCNT     TRUE IF THE DAYTIME CONTOLLER OPERATES
C             FALSE IF THE DAYTIME CONTROLLER DOES NOT OPERATE
C  DELY       TEMPERATURE DIFFERENCE
```

```
C  EPS        POSITIVE SMALL NUMBER FOR ACCURACY USED IN ROOT
C  EPSOFF     POSITIVE SMALL NUMBER
C  EPSON      POSITIVE SMALL NUMBER
C  EPSSET     POSITIVE SMALL NUMBER
C  EPSX       TOLERANCE DETERMINED BY SAMPLING FREQUENCY
C  FSYSON     TRUE IF THE SYSTEM OPERATES AT FULL POWER
C             FALSE IF THE SYSTEM IS OFF
C  HEAT       TRUE FOR HEATING MODE
C             FALSE FOR COOLING MODE
C  ICYCLE     NUMBER OF  CYCLES (ONE CYCLE PER DAY)
C  ITDOWN     INDEX NUMBER WHERE TEMPERATURE DECAY BEGINS
C  ITIME      INDEX NUMBER OF MODIFIED TIME OF DAY
C  ITMAX      MAXIMUM NUMBER OF ITERATIONS TO OBTAIN A ROOT USED IN ROOT
C  ITOFF      INDEX NUMBER WHERE X=XOFF
C  ITON       INDEX NUMBER WHERE X=XSTART
C  ITSET      INDEX NUMBER WHERE TEMPERATURE REACHES THE SET POINT
C  ITUNOC     INDEX NUMBER WHERE X=XUNOC
C  ITUP       INDEX NUMBER WHERE TEMPERATURE RISE BEGINS
C  MAINTN     TRUE IF THE ON-OFF CONTROL IS NEEDED
C             FALSE IF THE ON-OFF CONTROL IS NOT NEEDED
C  NMAX       TOTAL NUMBER OF SAMPLES A DAY
C  NPTHR      NUMBER OF SAMPLES IN ONE HOUR
C  PARTON     TRUE IF THE SYSTEM OPERATES WITH A PARTIAL CAPACITY
C             FALSE IF THE SYSTEM IS OFF
C  SWI        EVENT CONTROL SWITCHES (I=0,...,7)
C  TAUOFF     SYSTEM TIME CONSTANT DURING THE OFF-PERIOD
C  TAUON      SYSTEM TIME CONSTANT DURING THE ON-PERIOD
C  TBEGIN     ORIGIN OF X-COORDINATE (MILITARY TIME)
C  TDEDOF     DEAD TIME DURING THE OFF-PERIOD
C  TDEDON     DEAD TIME DURING THE ON-PERIOD
C  TEMP       INDOOR OR INTERIOR SURFACE TEMPERATURE
C  TIME       TIME OF DAY (MILITARY TIME)
C  TIMEX      MODIFIED TIME OF DAY IN SCALAR
C  TOA(*)     OUTDOOR TEMPERATURE
C  TOCC       BEGINING TIME OF OCCUPANCY (MILITARY TIME)
C  TOFF       OPTIMUM STOP TIME (MILITARY TIME)
C  TOUT       DRY-BULB OUTDOOR TEMPERATURE
C  TSTART     OPTIMUM START TIME (MILITARY TIME)
C  TUNOC      BEGINING TIME OF UNOCCUPANCY (MILITARY TIME)
C  X(I)       MODIFIED TIME OF DAY WITH ITS ORIGIN AT TBEGIN
C  XD         X-VALUE WHERE Y-VALUE IS EQUAL TO YSET
C  XDATA(*)   MODIFIED TIME OF DAY (X-VALUE)
C  XOCC       BEGINNING TIME OF OCCUPANCY IN X-COORDINATE
C  XOCC1      PAST VALUE OF XOCC
C  XOFF       OPTIMUM STOP TIME IN X-COORDINATE
C  XSTART     OPTIMUM START TIME IN X-COORDINATE
C  XUNOC      BEGINNING TIME OF UNOCCUPANCY IN X-COORDINATE
C  XUNOC1     PAST VALUE OF XOCC
C  Y(I)       TEMPERATURE
C  YDATA(*)   TEMPERATURE (Y-VALUE)
C  YINF       STEADY-STATE TEMPERATURE AS X GOES TO INFINITY DURING
```

```
C              THE OFF-PERIOD
C  YMAX        MAXIMUM TEMPERATURE
C  YMIN        MINIMUM TEMPERATURE
C  YSET        SET POINT TEMPERATURE (TARGET TEMPERATURE)
C  YSS         STEADY-STATE TEMPERATURE AS X GOES TO INFINITY DURING
C              THE ON-PERIOD
C
C ***************************************************************
C
      SUBROUTINE OPTSS(TIME,TEMP,TOUT)
      LOGICAL SW0,SW1,SW2,SW3,SW4,SW5,SW6,SW7,HEAT,MAINTN,PARTON,
     & FSYSON,DAYCNT
      LOGICAL ALARM                                  ;*** ADDED 8-13-84 ***
      LOGICAL OPENED,WARMST                          ;*** ADDED 8-24-84 ***
      LOGICAL RETRY                                 ;*** ADDED 10-01-84 ***
      EXTERNAL FYDIF
C-----PARAMETER (NPTHR=4,NMAX=NPTHR*24)-----original statement replaced by:
      PARAMETER (NPTHRX=60,NMAX=NPTHRX*24) ;*** 7-27-84 NPTHRX is max NPTHR
      DIMENSION XDATA(0:NMAX),YDATA(0:NMAX),TOA(0:NMAX)
      COMMON /XY/ X(4),Y(4),XOCC,XUNOC,YSET
     &       /CNST/ TAUON,TAUOFF,YINF,YSS,XD
     &       /LIMIT/ YMIN,YMAX,HEAT,MAINTN,FSYSON,DAYCNT,XSTART
     &       /TSET/ TSTART,TOFF,TOCC,TUNOC,TBEGIN
      COMMON /CLIM/ TUNOMX                          ;*** ADDED 10-01-84 ***
      COMMON /FREQ/ NPTHR                           ;*** ADDED 7-27-84 ***
C-----COMMON /MSG/ MLU ----------------------------;*** ADDED 7-27-84 ***
      COMMON /MSG/ MLU,ALARM                        ;*** ADDED 8-13-84 ***
      COMMON /EPSLON/ EPS,EPSX,EPSOFF,EPSON,EPSSET;*** ADDED 8-02-84 ***
      NAMELIST /RESULT/X,Y,XOFF,XSTART,TAUON,TAUOFF,TDEDON,TDEDOF,
     &    YSS,YINF,ICYCLE,ITIME,ITOFF,ITUNOC,ITDOWN,ITON,ITUP,ITSET,
     &    TBEGIN,TOFF,TUNOC,TSTART,TOCC
C*****************************************************************************
      NAMELIST/DEBUG/TAUOFF,YINF,OFFL,OFFR,ONL,ONR,IFLAG,ITER,HEAT   ;debugging
C*****************************************************************************
C-----NAMELIST /SAVE/XOFF,X,Y,XSTART,TOAMAX,TOAMIN,XLEFT,XRIGHT,--------
C----&TDEDON,TBEGIN,YSS----;*** ADDED 8-24-84 *** REPLACED 8-30-84 ***--
      NAMELIST /SAVE/TBEGIN,TDEDOF,XOFF,ITDOWN,X,Y,XSTART,TOAMAX,TOAMIN,
     &XLEFT,XRIGHT,ITON,TDEDON,ITUP,ITSET,YSS,YINF ;*** ADDED 8-30-84 ***
C
C-----DATA EPS,ITMAX/0.005,20/-----------------*** REMOVED 8-02-84 ***
C-----DATA SW0/.TRUE./,EPSOFF,EPSON,EPSSET/3*0.25/*** REMOVED 8-02-84 ***
      DATA ITMAX/50/,SW0/.TRUE./                    ;*** ADDED 8-02-84 ***
C
C     INITIAL CONDITIONS
C
C-----EPSX=0.5/NPTHR-----------------------------*** REMOVED 8-02-84 ***
      IF(SW0) THEN
        ICYCLE=0
        SW1=.FALSE.
        SW2=.FALSE.
        SW3=.FALSE.
```

```
        SW4=.FALSE.
        SW5=.FALSE.
        SW6=.FALSE.
        SW7=.FALSE.
        MAINTN=.FALSE.
        PARTON=.FALSE.
        FSYSON=.FALSE.
        DAYCNT=.TRUE.
        ALARM=.FALSE.                                  ;*** ADDED 8-13-84 ***
C
C     CHANGE TIME COORDINATES
C
        CALL XCOORD(XSTART,TSTART,TBEGIN,1)
        CALL XCOORD(XOCC,TOCC,TBEGIN,1)
        CALL XCOORD(XUNOC,TUNOC,TBEGIN,1)
        CALL XCOORD(XUNOMX,TUNOMX,TBEGIN,1)       ;*** ADDED 10-01-84 ***
C
C     TRY TO INITIALIZE PARAMETERS                 ;*** ADDED 8-24-84 ***
C
        CALL OPSAVE(OPENED,'OLD    ')              ;*** ADDED 8-24-84 ***
        IF(OPENED)THEN                             ;*** ADDED 8-24-84 ***
          READ(2,SAVE)                             ;*** ADDED 8-24-84 ***
          CLOSE(2)                                 ;*** ADDED 8-24-84 ***
          SW0=.FALSE.                              ;*** ADDED 8-29-84 ***
          SW1=.TRUE.                               ;*** ADDED 8-24-84 ***
          ICYCLE=1                                 ;*** ADDED 8-24-84 ***
          WARMST=.TRUE.                            ;*** ADDED 8-24-84 ***
          WRITE(MLU,FMT="(1X,'PARAMETERS TAKEN FROM FILE')");***ADDED 8-24-84
        ELSE                                       ;*** ADDED 8-24-84 ***
          WARMST=.FALSE.                           ;*** ADDED 8-24-84 ***
        ENDIF                                      ;*** ADDED 8-24-84 ***
C
      ENDIF
C
C     STORE INPUT DATA AS ARRAYS IN TERMS OF INDEX TIME
C
      CALL XCOORD(TIMEX,TIME,TBEGIN,1)
      ITIME=TIMEX*NPTHR
      TOA(ITIME)=TOUT
      XDATA(ITIME)=TIMEX
      YDATA(ITIME)=TEMP
      X(4)=XDATA(ITIME)
      Y(4)=YDATA(ITIME)
C
C     TURN OFF THE SYSTEM AT THE BEGINNING OF UNOCCUPANCY OF INITIAL
C     CYCLE
C
      IF(SW0.AND.ABS(XDATA(ITIME)-XUNOC).LT.EPSX) THEN
        DAYCNT=.FALSE.
        ITOFF=ITIME
        ITUNOC=ITIME
```

```
        SW0=.FALSE.
        SW2=.TRUE.
        WRITE(MLU,FMT="(5X,'---SW0---SYSTEM OFF AT THE FIRST CYCLE')")
      ENDIF
C
C     TURN OFF THE SYSTEM AT THE BEGINNING OF UNOCCUPANCY AFTER INITIAL
C     CYCLE
C
      IF(SW1.AND.XDATA(ITIME).GE.XOFF) THEN
        DAYCNT=.FALSE.
        ITOFF=ITIME
        SW1=.FALSE.
        SW2=.TRUE.
        WRITE(MLU,FMT="(5X,'---SW1---SYSTEM OFF')")
      ENDIF
C
C     DETERMINE DEAD TIME DURING THE OFF-PERIOD
C
      IF(SW2.AND.ABS(YDATA(ITIME)-YDATA(ITOFF)).GE.EPSOFF) THEN
        TDEDOF=XDATA(ITIME)-XDATA(ITOFF)
        XOFF=XUNOC-TDEDOF
        IF(XOFF.LT.XUNOMX)XOFF=XUNOMX              ;*** ADDED 10-01-84 ***
        ITDOWN=ITIME
        SW2=.FALSE.
        SW3=.TRUE.
        SW4=.TRUE.
        WRITE(MLU,FMT="(5X,'---SW2---DECAY/RISE RESPONSE OCCURS')")
      ELSE IF(SW2.AND.ALARM) THEN                  ;*** ADDED 8-13-84 ***
        SW2=.FALSE.                                ;*** ADDED 8-13-84 ***
        SW3=.FALSE.                                ;*** ADDED 8-13-84 ***
        SW4=.TRUE.                                 ;*** ADDED 8-13-84 ***
        ALARM=.FALSE.                              ;*** ADDED 8-13-84 ***
      ENDIF
C
C     MAINTAIN MINIMUM LEVEL OF OPERATION
C
C-----IF(SW3) THEN-----------------------------;*** REPLACED 8-29-31 ***
      IF(SW3.AND.(.NOT.DAYCNT)) THEN               ;*** ADDED 8-29-84 ***
        IF((HEAT.AND.(Y(4).LE.YMIN)).OR.(.NOT.(HEAT).AND.
     &  Y(4).GT.YMAX)) THEN
          IF(ICYCLE.GE.1) XSTART=XLEFT-TDEDON
          MAINTN=.TRUE.
          SW3=.FALSE.
          WRITE(MLU,FMT="(5X,'---SW3---ON-OFF CONTROLLER TAKES OVER')")
        ENDIF
      ENDIF
C
C     FIND THE INTERSECTION OF ON- AND OFF-TEMPERATURE CURVES
C
      IF(ICYCLE.GE.1) THEN
        IF(ABS(XDATA(ITIME)-X(3)).LT.EPSX) THEN
```

```
          Y(3)=YDATA(ITIME)
        ENDIF
C
C-------IF((.NOT.MAINTN).AND.(XDATA(ITIME).GT.X(3).AND.XDATA(ITIME)
C----&   .LT.XSTART).AND.(ABS(Y(3)-Y(4)).GT.EPSSET)) THEN;*** REMOVED 9-27-84 *
        IF((.NOT.MAINTN).AND.(XDATA(ITIME).GT.X(3).AND.XDATA(ITIME)
     &   .LT.XSTART)) THEN                                ;** ADDED 9-27-84 ***
C
C---------IF(HEAT) THEN-----------------------------;*** REMOVED 9-27-84 ***
            IF(Y(4).GE.Y(3)) THEN
              YINF=TOAMAX
            ELSE
              YINF=TOAMIN
            ENDIF
C---------ELSE---------------------------------------;*** REMOVED 9-27-84 ***
C----------IF(Y(4).LE.Y(3)) THEN--------------------;*** REMOVED 9-27-84 ***
C------------YINF=TOAMIN-------------------------;*** REMOVED 9-27-84 ***
C----------ELSE---------------------------------;*** REMOVED 9-27-84 ***
C------------YINF=TOAMAX-------------------------;*** REMOVED 9-27-84 ***
C----------ENDIF---------------------------------;*** REMOVED 9-27-84 ***
C---------ENDIF----------------------------------;*** REMOVED 9-27-84 ***
C
          XL=XLEFT
          XR=XRIGHT
          RETRY = .FALSE.                          ;** ADDED 10-01-84 **
C---------CALL ROOT(XL,XR,EPS,ITMAX,FYDIF,XROOT,IFLAG);*** REPLACED 8-14-84 **
          CALL ROOT(XL,XR,EPS,ITMAX,FYDIF,XROOT,IFLAG,ITER)
C*************************************************
          OFFL=YOFF(XL)                  ;debugging *
          OFFR=YOFF(XR)                  ; 8-1-84   *
          ONL=YONNEW(XL)                 ;          *
          ONR=YONNEW(XR)                 ;          *
          WRITE(MLU,DEBUG)               ;          *
C*************************************************
          IF(IFLAG.NE.1) THEN
            XSTART=XROOT-TDEDON
          ELSE
            IF(.NOT.RETRY) THEN                    ;** ADDED 10-01-84 **
              RETRY = .TRUE.                       ;** ADDED 10-01-84 **
              CALL CHMODE(HEAT,YSS,YSET)           ;** ADDED 10-01-84 **
              CALL ROOT(XL,XR,EPS,ITMAX,FYDIF,XROOT,IFLAG,ITER) ;ADDED 10-1-84 *
C*************************************************
          OFFL=YOFF(XL)                  ;debugging *
          OFFR=YOFF(XR)                  ; 8-1-84   *
          ONL=YONNEW(XL)                 ;          *
          ONR=YONNEW(XR)                 ;          *
          WRITE(MLU,DEBUG)               ;          *
C*************************************************
              IF(IFLAG.NE.1) THEN                  ;** ADDED 10-01-84 **
                XSTART = XROOT - TDEDON            ;** ADDED 10-01-84 **
              ELSE                                 ;** ADDED 10-01-84 **
```

```
            CALL CHMODE(HEAT,YSS,YSET)                ;** ADDED 10-01-84 **
          ENDIF                                       ;** ADDED 10-01-84 **
        ENDIF                                         ;** ADDED 10-01-84 **
        IF(RETRY.AND.IFLAG.EQ.1) THEN                 ;** ADDED 10-01-84 **
           XSTART = XLEFT-TDEDON
        ENDIF                                         ;** ADDED 10-01-84 **
      ENDIF
    ENDIF
    CALL XCOORD(XSTART,TSTART,TBEGIN,2)
  ENDIF
C
C     TRUN ON THE SYSTEM AT OPTIMUM START TIME
C
  IF(SW4.AND.XDATA(ITIME).GE.XSTART) THEN
    FSYSON=.TRUE.
    ITON=ITIME
    MAINTN=.FALSE.
    SW4=.FALSE.
    SW5=.TRUE.
    WRITE(MLU,FMT="(5X,'---SW4---SYSTEM ON AT FULL POWER')")
  ENDIF
C
C     DETERMINE DEAD TIME DURING THE ON-PERIOD
C
  IF(SW5.AND.ABS(YDATA(ITIME)-YDATA(ITON)).GE.EPSON) THEN
    TDEDON=XDATA(ITIME)-XSTART
    ITUP=ITIME
    SW5=.FALSE.
    SW6=.TRUE.
    WRITE(MLU,FMT="(5X,'---SW5---UPRISE/DECAY RESPONSE OCCURS')")
  ELSE IF(SW5.AND.ALARM) THEN                     ;*** ADDED 8-13-84 ***
    FSYSON=.FALSE.                                ;*** ADDED 8-13-84 ***
    DAYCNT=.TRUE.                                 ;*** ADDED 8-13-84 ***
    SW5=.FALSE.                                   ;*** ADDED 8-13-84 ***
    SW7=.FALSE.                                   ;*** ADDED 8-13-84 ***
    ALARM=.FALSE.                                 ;*** ADDED 8-13-84 ***
  ENDIF
C
C     TURN OFF THE SYSTEM NEAR THE BEGINNING OF OCCUPANCY AND CALL
C     THE DAYTIME CONTROLLER
C
  IF(SW6.AND.((ABS(YDATA(ITIME)-YSET).LE.EPSSET)
 &  .OR.(HEAT.AND.YDATA(ITIME).GE.YSET)
 &  .OR.(.NOT.HEAT.AND.YDATA(ITIME).LE.YSET))) THEN
    FSYSON=.FALSE.
    DAYCNT=.TRUE.
    ITSET=ITIME
    SW6=.FALSE.
    SW7=.TRUE.
    WRITE(MLU,FMT="(5X,'---SW6---DAYTIME CONTROLLER TAKES OVER')")
  ELSE IF(SW6.AND.ALARM) THEN                     ;*** ADDED 8-13-84 ***
```

```
        FSYSON=.FALSE.                                  ;*** ADDED 8-13-84 ***
        DAYCNT=.TRUE.                                   ;*** ADDED 8-13-84 ***
        SW6=.FALSE.                                     ;*** ADDED 8-13-84 ***
        SW7=.FALSE.                                     ;*** ADDED 8-13-84 ***
        ALARM=.FALSE.                                   ;*** ADDED 8-13-84 ***
      ENDIF
C
C     COMPUTE X- AND Y-VALUES BASED ON PREVIOUS OBSERVATION
C      AND UPDATE THEM USING NEW TOCC AND TUNOC
C
      IF(SW7) THEN
        CALL XCOORD(XOFF,TOFF,TBEGIN,2)
        WRITE(MLU,900)
        WRITE(MLU,RESULT)
        WRITE(MLU,900)
        XOCC1=XOCC
        XUNOC1=XUNOC
        IF(TUNOC.GT.TOCC) THEN
          TBEGIN=0.5*(TUNOC+TOCC)
        ELSE
          TBEGIN=0.5*(TUNOC+24.+TOCC)
          IF(TBEGIN.GE.24.) TBEGIN=TBEGIN-24.
        ENDIF
        CALL XCOORD(XOCC,TOCC,TBEGIN,1)
        CALL XCOORD(XUNOC,TUNOC,TBEGIN,1)
        CALL XCOORD(XUNOMX,TUNOMX,TBEGIN,1)      ;*** ADDED 10-01-84 ***
        K1=(ITSET-ITUP)/3+ITUP
        K2=(ITSET-ITUP)*2/3+ITUP
        IF(ICYCLE.EQ.0) THEN
          K3=(ITON-ITUNOC)/4+ITUNOC
          ISTART=XUNOC1*NPTHR                       ;*** ADDED 8-02-84 ***
          IEND=TIMEX*NPTHR                          ;*** ADDED 8-02-84 ***
        ELSE
          K3=(ITON-ITDOWN)/4+ITDOWN
          ISTART=XUNOC*NPTHR                        ;*** ADDED 8-02-84 ***
          IEND=XOCC*NPTHR                           ;*** ADDED 8-02-84 ***
        ENDIF
        X(1)=XDATA(K1)+XOCC-XOCC1
        X(2)=XDATA(K2)+XOCC-XOCC1
        X(3)=XDATA(K3)+XUNOC-XUNOC1
        Y(1)=YDATA(K1)
        Y(2)=YDATA(K2)
        Y(3)=YDATA(K3)
        XSTART=XSTART+XOCC-XOCC1
        XOFF=XOFF+XUNOC-XUNOC1
        CALL XCOORD(XSTART,TSTART,TBEGIN,2)
        CALL XCOORD(XOFF,TOFF,TBEGIN,2)
        CALL XCOORD(XOFF,TOFF,TBEGIN,1)           ;** ADDED 10-01-84 **
C
C     DETERMINE MAXIMUM AND MINIMUM OUTDOOR TEMPERATURES
C
```

```
C-------CALL MAXMIN(TOA,TOAMIN,TOAMAX,NMAX)------*** REMOVED 8-02-84 ***
C-------CALL MAXMIN(TOA,TOAMIN,TOAMAX,ISTART,IEND);*** ADDED 8-02-84 ***
        IF(.NOT.WARMST)
     &  CALL MAXMIN(TOA,TOAMIN,TOAMAX,ISTART,IEND);*** ADDED 8-24-84 ***
C
C     COMPUTE CONSTANTS FOR ON- AND OFF-PERIOD TEMPERATURE EQUATIONS
C
        IF(HEAT) THEN
          YSS=1.5*YSET
          YINF=TOAMIN
        ELSE
          YSS=0.5*YSET
          YINF=TOAMAX
        ENDIF
C
C     DETERMINE XLEFT AND XRIGHT
C
        CALL XLTRT(XLEFT,XRIGHT)
        ICYCLE=ICYCLE+1
        SW7=.FALSE.
        WRITE(MLU,FMT="(5X,'--SW7--CALCULATIONS FOR NEXT CYCLE DONE')")
        WRITE(MLU,900)
        WRITE(MLU,RESULT)
        WRITE(MLU,900)
C
C     SAVE PARAMETERS IN FILE FOR WARM START      ;*** ADDED 8-24-84 ***
C
        CALL OPSAVE(OPENED,'RENEW ')              ;*** ADDED 8-24-84 ***
        IF(OPENED)THEN                            ;*** ADDED 8-24-84 ***
          WRITE(2,SAVE)                           ;*** ADDED 8-24-84 ***
          CLOSE(2)                                ;*** ADDED 8-24-84 ***
        ELSE                                      ;*** ADDED 8-24-84 ***
          WRITE(MLU,FMT="(1X,'PARAMETERS NOT SAVED IN FILE')");***ADDED 8-24-84
        ENDIF                                     ;*** ADDED 8-24-84 ***
      ENDIF
C
C       RESET SWITCH SW1 AT THE BEGINNING OF NEXT CYCLE
C
      IF(ICYCLE.GE.1.AND.ABS(TIME-TBEGIN).LT.EPSX) SW1=.TRUE.
C
C      FORMAT STATEMENTS
C
900   FORMAT(80('*')/)
C
C
      RETURN
      END
C ***************************************************************************
C
C     XCOORD : X-COORDINATE TRANSFORM
C
```

```
C     IFLAG   =1 FOR TRANSFORM OF TIME OF DAY INTO X-VALUE
C             =2 FOR TRANSFORM OF X-VALUE INTO TIME OF DAY
C
C ***************************************************************
C
      SUBROUTINE XCOORD(X,TIME,TBEGIN,IFLAG)
      IF(IFLAG.EQ.1) THEN
        IF(TIME.GE.TBEGIN.AND.TIME.LE.24.)   THEN
          X=TIME-TBEGIN
        ELSE
          X=TIME+(24.-TBEGIN)
        ENDIF
      ELSE
        TIME=X+TBEGIN
        IF(TIME.GE.24.0) TIME=TIME-24.0
      ENDIF
C
      RETURN
      END
C ***************************************************************
C
C       FYDIF : TEMPERATURE DIFFERENCE
C
C -------------------------------------------------------------
C
C     YONNEW  UPDATED TEMPERATURE  DURING THE HEAT-UP PERIOD
C     YOFF    TEMPERATURE  DURING THE COOL-DOWN PERIOD
C
C ***************************************************************
C
      FUNCTION FYDIF(XX)
C
C       TEMPERATURE DIFFERENCE BETWEEN YONNEW AND YOFF
C       TO BE USED IN DETERMINATION OF INTERSECTION, XROOT
C
      FYDIF=YONNEW(XX)-YOFF(XX)
      RETURN
      END
C ***************************************************************
C
C     YONNEW, YOFF : THE ON- AND OFF-PERIOD TEMPERATURES
C
C ***************************************************************
C
      FUNCTION YONNEW(XX)
      LOGICAL WRZERU,WRZERD                          ;*** ADDED 7-30-84 ***
      COMMON /XY/ X(4),Y(4),XOCC,XUNOC,YSET
     &       /CNST/ TAUON,TAUOFF,YINF,YSS,XD
      COMMON /MSG/ MLU, ALARM ;*** ADDED 7-27-84 AND CHANGED 8-13-84 ***
      DATA WRZERU/.TRUE./,WRZERD/.TRUE./             ;*** ADDED 7-30-84 ***
      PARAMETER (EPSTAU=0.033)                       ;*** ADDED 7-27-84 ***
```

```
C
C     THE ON-PERIOD TEMPERATURE WITH ADJUSTMENT
C
      ARGLOG = (YSS-Y(1))/(YSS-Y(2))                    ;*** ADDED 9-27-84 ***
      IF(ARGLOG.GT.0.0.AND.ARGLOG.NE.1)THEN             ;*** ADDED 9-27-84 ***
        TAUON=(X(2)-X(1))/ALOG(ARGLOG)                  ;*** ADDED 9-27-84 ***
C-------TAUON=(X(2)-X(1))/ALOG((YSS-Y(1))/(YSS-Y(2))) ;*** REMOVED 9-27 ***
      ELSE                                              ;*** ADDED 9-27-84 ***
        TAUON = 0.0                                     ;*** ADDED 9-27-84 ***
      ENDIF
      IF(TAUON.LT.EPSTAU)THEN                           ;*** ADDED 7-27-84 ***
       TAUON=EPSTAU                                     ;*** ADDED 7-27-84 ***
       IF(WRZERU)THEN                                   ;*** ADDED 7-30-84 ***
        WRZERU=.FALSE.                                  ;*** ADDED 7-30-84 ***
       WRITE(MLU,FMT="(1X,'*** WARNING *** The start-up time constant is
     > zero.')")                                        ;*** ADDED 7-27-84 ***
       ENDIF                                            ;*** ADDED 7-30-84 ***
      ELSE                                              ;*** ADDED 7-30-84 ***
       WRZERU=.TRUE.                                    ;*** ADDED 7-30-84 ***
      ENDIF                                             ;*** ADDED 7-27-84 ***
      XD=X(1)-TAUON*ALOG((YSS-YSET)/(YSS-Y(1)))
      X1P=X(1)+XOCC-XD
      YONNEW=YSS-(YSS-Y(1))*EXP((X1P-XX)/TAUON)
      RETURN
C
      ENTRY YOFF(XX)
C
C     THE OFF-PERIOD TEMPERATURE
C
      IF(Y(3).NE.Y(4))THEN                              ;*** ADDED 9-27-84 ***
        TAUOFF=(X(4)-X(3))/ALOG((Y(3)-YINF)/(Y(4)-YINF))
      ELSE                                              ;*** ADDED 9-27-84 ***
        TAUOFF = 0.0                                    ;*** ADDED 9-27-84 ***
      ENDIF
      IF(TAUOFF.LT.EPSTAU)THEN                          ;*** ADDED 7-27-84 ***
       TAUOFF=EPSTAU                                    ;*** ADDED 7-27-84 ***
       YOFF = Y(4)                                      ;*** ADDED 9-27-84 ***
       IF(WRZERD)THEN                                   ;*** ADDED 7-30-84 ***
        WRZERD=.FALSE.                                  ;*** ADDED 7-30-84 ***
       WRITE(MLU,FMT="(1X,'*** WARNING *** The shut-down time constant i
     >s zero.')")                                       ;*** ADDED 7-27-84 ***
       ENDIF                                            ;*** ADDED 7-30-84 ***
      ELSE                                              ;*** ADDED 7-30-84 ***
       WRZERD=.TRUE.                                    ;*** ADDED 7-30-84 ***
C-----ENDIF--------------;*** ADDED 7-27-84 ***         ;*** REMOVED 9-27-84 ***
       YOFF=(Y(3)-YINF)*EXP((X(3)-XX)/TAUOFF)+YINF
      ENDIF                                             ;*** ADDED 9-27-84 ***
      RETURN
      END
C ***************************************************************************
C
```

```
C       ROOT : FIND A ROOT OF A FUNCTION F(X)=0 IN A GIVEN INTERVAL
C              BY THE REGULA-FALSI METHOD
C
C     REFERENCE :
C          CARNAHAN, LUTHER, AND WILKES
C          " APPLIED NUMERICAL METHODS ", JOHN WILEY, 1969, P.193
C
C     THE ORIGINAL PROGRAM WAS MODIFIED IN FORTRAN77.
C
C ---------------------------------------------------------------
C
C     EPS      A POSITIVE SMALL NUMBER FOR ACCURACY
C     FX       SCALAR FUNCTION VALUE
C     IFLAG    0 IF FINDING A ROOT IS SUCCESSFUL
C              1 IF FINDING A ROOT IS NOT SUCCESSFUL
C              2 IF MAXIMUM ITERATIONS EXCEEDED     *** ADDED 8-1-84 ***
C     ITMAX    MAXIMUM NUMBER OF ITERATIONS
C     XL       LEFT-MOST X-VALUE
C     XR       RIGHT-MOST X-VALUE
C     XROOT    A REAL ROOT
C
C ***************************************************************************
C
C-----SUBROUTINE ROOT(XL,XR,EPS,ITMAX,FX,XROOT,IFLAG)** REPLACED 8-13-84
      SUBROUTINE ROOT(XL,XR,EPS,ITMAX,FX,XROOT,IFLAG,ITER)
C
      COMMON /MSG/ MLU, ALARM ;*** ADDED 7-27-84 AND CHANGED 8-13-84 ***
C
C     SET LEFTMOST AND RIGHTMOST FUNCTION VALUES
C
      FXL=FX(XL)
      FXR=FX(XR)
C
C     CHECK FOR PRESENCE OF A ROOT
C
      IFLAG=0                              ;*** MOVED FROM BELOW 8-1-84 ***
      IF(FXL*FXR.LT.0.) THEN
C
C       BEGIN REGULA FALSI ITERATION
C
        DO 10 ITER=1,ITMAX
        X2=(XL*FXR-XR*FXL)/(FXR-FXL)
        FX2=FX(X2)
C
C       CHECK FOR CONVERGENCE
C
        IF(ABS(FX2).LE.EPS) GOTO 20
C
C       KEEP RIGHT OR LEFT SUBINTERVAL
C
        IF(FX2*FXL.LT.0.) THEN
```

```
          XR=X2
          FXR=FX2
        ELSE
          XL=X2
          FXL=FX2
        ENDIF
10      CONTINUE
        IFLAG=2                                          ;*** ADDED 8-1-84 ***
      ELSEIF (FXL*FXR.EQ.0.) THEN
        ITER=1
        IF(FXL.EQ.0.) THEN
          X2=XL
          FX2=0.
        ELSE
          X2=XR
          FX2=0.
        ENDIF
      ELSE
        IFLAG=1
        RETURN
      ENDIF
20    XROOT=X2
CCC---IFLAG=0--------------------THIS LINE MOVED TO BEGINNING 8-1-84---
C
      RETURN
      END
C ***************************************************************************
C
C     XLTRT :  LEFTMOST AND RIGHTMOST X-VALUES FOR THE SUBROUTINE, ROOT
C
C -------------------------------------------------------------
C
C     DX       TIME INCREMENT
C     N        NUMBER OF INTERVALS
C     XL       LEFT-MOST VALUE OF X
C     XR       RIGHT-MOST VALUE OF X
C
C ***************************************************************************
C
      SUBROUTINE XLTRT(XL,XR)
      LOGICAL HEAT
      COMMON /XY/ X(4),Y(4),XOCC,XUNOC,YSET
     &      /LIMIT/ YMIN,YMAX,HEAT,MAINTN,FSYSON,DAYCNT,XSTART
      COMMON /MSG/ MLU, ALARM ;*** ADDED 7-27-84 AND CHANGED 8-13-84 ***
C
C     TIME BASIS  IS HOUR, AND TIME INCREMENT IS 5 MIN.
C
      DX=5./60.
      N=(XOCC-XUNOC)/DX
      XX=X(2)
      DO 10 I=1,N
```

```
      IF(HEAT) THEN
         IF(YONNEW(XX).LE.YMIN) GOTO 20
      ELSEIF(YONNEW(XX).GE.YMAX) THEN
         GOTO 20
      ENDIF
      XX=XX-DX
10    CONTINUE
20    XL=XX
      XR=XOCC
      WRITE(MLU,FMT="(5X,'XLEFT=',F10.4,5X,'XRIGHT=',F10.4/)")XL,XR
C
      RETURN
      END
C ***************************************************************
C
C       MAXMIN : EVALUATION OF MAXIMUM AND MINIMUM VALUES
C
C ---------------------------------------------------------------
C
C     A       ARRAY VALUES
C     AMAX    MAXIMUM VALUE OF ARRAY VALUES
C     AMIN    MIMINUM VALUE OF ARRAY VALUES
C     IEND    INDEX NUMBER OF ENDING OF SEARCH
C     ISTART  INDEX NUMBER OF START OF SEARCH
C
C ***************************************************************
C
C------SUBROUTINE MAXMIN(A,AMIN,AMAX,NMAX)-------*** REMOVED 8-02-84 ***
      SUBROUTINE MAXMIN(A,AMIN,AMAX,ISTART,IEND) ;*** ADDED 8-02-84 ***
C------DIMENSION A(0:NMAX)-----------------------*** REMOVED 8-02-84 ***
      DIMENSION A(0:*)                           ;*** ADDED 8-02-84 ***
C------COMMON /XY/X(4),Y(4),XOCC,XUNOC,YSET------*** REMOVED 7-27-84 ***
C***************************************************************
        NAMELIST/DEBUG/AMIN,AMAX,ISTART,IEND
C***************************************************************
C------NPTHR=NMAX/24------------------------------*** REMOVED 7-27-84 ***
C------ISTART=XUNOC*NPTHR-------------------------*** REMOVED 8-02-84 ***
C------IEND=XOCC*NPTHR----------------------------*** REMOVED 8-02-84 ***
C------AMIN=A(ISTART-1)---------------------------*** REMOVED 8-02-84 ***
C------AMAX=A(ISTART-1)---------------------------*** REMOVED 8-02-84 ***
      AMIN=A(ISTART)                             ;*** ADDED 8-02-84 ***
      AMAX=A(ISTART)                             ;*** ADDED 8-02-84 ***
      DO 10 I=ISTART,IEND
      IF(A(I).LT.AMIN) AMIN=A(I)
      IF(A(I).GT.AMAX) AMAX=A(I)
10    CONTINUE
C***************************************************************
        WRITE(1,DEBUG)
C***************************************************************
C
      RETURN
```

```
      END
C ***********************************************************************
C
C     ONOFF : THE ON-OFF CONTROL ALGORITHM
C
C ***********************************************************************
C
      SUBROUTINE ONOFF(TEMP)
      LOGICAL HEAT,MAINTN,FSYSON,DAYCNT,PARTON
      COMMON /LIMIT/ YMIN,YMAX,HEAT,MAINTN,FSYSON,DAYCNT,XSTART
     &       /ONOFFC/ PARTON,DELY
C
      IF(HEAT) THEN
        IF(TEMP.LE.YMIN) PARTON=.TRUE.
        IF(TEMP.GE.YMIN+DELY) PARTON=.FALSE.
      ELSEIF(TEMP.GE.YMAX) THEN
        PARTON=.TRUE.
      ELSEIF(TEMP.LE.YMAX-DELY) THEN
        PARTON=.FALSE.
      ENDIF
C
      RETURN
      END
C=======================================================================
      SUBROUTINE CHMODE(HEAT,YSS,YSET)
C=======================================================================
C This routine is used to flip-flop modes from heating mode to cooling
C mode and vice-versa. The steady state start-up temperature, YSS, is
C also calculated. This entire routine ** ADDED 10-01-84 **
      LOGICAL HEAT
      REAL YSS,YSET
C---------flip mode-------------------------------------------------------
      IF(HEAT) THEN
        HEAT = .FALSE.
      ELSE
        HEAT = .TRUE.
      ENDIF
C--------redetermine YSS-------------------------------------------------
      IF(HEAT) THEN
        YSS=1.5*YSET
      ELSE
        YSS=0.5*YSET
      ENDIF
      RETURN
      END
C=======================================================================
C
C     OPTEST:  TEST VERSION OF MAIN PROGRAM OF OPTIMUM START/STOP CONTROL
C            FOR BOTH HEATING AND COOLING SEASONS
C
C=======================================================================
```

```
      PROGRAM OPMAIN
C=====================================================================
      LOGICAL HEAT,MAINTN,FSYSON,DAYCNT,PARTON,WASDAY,INITDN,SO,DC
      LOGICAL ALARM
      INTEGER DAY,WRKDAY,HOLDAY
      COMMON/SELECT/IFIDC,IDTEMP,IDTOUT,TEMP,TOUT,NDELAY
      COMMON /XY/ X(4),Y(4),XOCC,XUNOC,YSET
     &       /CNST/ TAUON,TAUOFF,YINF,YSS,XD
     &       /LIMIT/ YMIN,YMAX,HEAT,MAINTN,FSYSON,DAYCNT,XSTART
     &       /ONOFFC/ PARTON,DELY
     &       /TSET/ TSTART,TOFF,TOCC,TUNOC,TBEGIN
      COMMON /CLIM/ TUNOMX                        ;*** ADDED 10-01-84 ***
      COMMON /FREQ/ NPTHR
      COMMON /MSG/ MLU,ALARM
      COMMON /EPSLON/ EPS,EPSX,EPSOFF,EPSON,EPSSET
      NAMELIST /OUTPUT/ TIME,TEMP,TOUT,SO,DC,TSTART,TOFF,TOCC,TUNOC,TB
      DATA WASDAY/.TRUE./,WRKDAY/1/,HOLDAY/0/
C---------------------------------------------------------------------
      MLU=1
      ILU=2
      OPEN(ILU,FILE='OPTSS.INI/50 ',RECL=80,SHARE='SRO',STATUS='OLD',
     >IOSTAT=IOPSTAT)
      IF(IOPSTAT.NE.0)THEN
      CALL CONMSG(48,'**ERROR** CANNOT FIND OPTSS INITIALIZATION FILE.')
       PAUSE
       GO TO 1000
      ENDIF
      READ(ILU,*)NPTHR
      READ(ILU,*)IDTEMP
      READ(ILU,*)IDTOUT
      READ(ILU,*)IFIDC
      READ(ILU,*)TOCC
      READ(ILU,*)TUNOC
      READ(ILU,*)TSTART
      READ(ILU,*)YSET
      READ(ILU,*)YMIN
      READ(ILU,*)YMAX
      READ(ILU,*)DELY
      READ(ILU,*)HEAT
      READ(ILU,*)EPS
      READ(ILU,*)EPSX
      READ(ILU,*)EPSOFF
      READ(ILU,*)EPSON
      READ(ILU,*)EPSSET
      READ(ILU,*)TUNOMX
      DAY=WRKDAY
      CLOSE(2)
C
      INITDN=.FALSE.
      CALL CONMSG(24,'INITIALIZATION COMPLETE.')
C---------open log file-------------------------------------------------
```

```
 1000 OPEN(MLU,FILE='OPTSS.DAT ',RECL=132,SHARE='SWO',IOSTAT=IOPSTAT)
      IF(IOPSTAT.NE.0)THEN
       CALL CONMSG(28,'**WARNING** CANNOT OPEN FILE')
       OPEN(MLU,FILE='C: ',RECL=80,SHARE='SWO',IOSTAT=IOPSTAT)
      ENDIF
C-----get space and outside air temperature and time-----------
      CALL OASTAT(TIME,TEMP,TOUT)
      IF(.NOT.INITDN)THEN
        TBEGIN=TIME
        INITDN=.TRUE.
      ENDIF
C------maintain minimum requirement during non-working days------
      IF(DAY.EQ.HOLDAY) THEN
        MAINTN=.TRUE.
        CALL ONOFF(TEMP)
      ELSE
        CALL OPTSS(TIME,TEMP,TOUT)
        IF((.NOT.WASDAY).AND.FSYSON)CALL AHU('START ')
        IF(WASDAY.AND.(.NOT.(DAYCNT.OR.FSYSON)))CALL AHU('STOP  ')
        WASDAY=(DAYCNT.OR.FSYSON)
      ENDIF
C*********************************************************************
        TB=TBEGIN
        SO=FSYSON
        DC=DAYCNT
        WRITE(1,OUTPUT)
C*********************************************************************
      CLOSE(MLU)
      CALL DELAY(NPTHR)
      GO TO 1000
      END
C=====================================================================
      SUBROUTINE DELAY(NPTHR)
C=====================================================================
      INTEGER HMS(3),SECOND
C----      NAMELIST /DEE/ DEL,PAST,SCHED,LEFT,HMS
      DATA SECOND/2/
C----------get number of seconds since start of hour ----------------
      RETURN
      END
C=====================================================================
      SUBROUTINE OASTAT(TIME,TEMPC,TOUTC)
C=====================================================================
      LOGICAL SO,DC
      INTEGER DEST
      INTEGER STATUS,FUNCTION,NDELAY,MS
      REAL*8 FID(8)
      LOGICAL IWAIT,IPAUSE,ICHANGE
      CHARACTER*122 NORESPONSE
      COMMON/CONTRO/IWAIT,IPAUSE,ICHANGE
      COMMON/SELECT/IFIDC,IDTEMP,IDTOUT,TEMP,TOUT,NDELAY
```

```
      DATA FID/'FID1    ','FID2    ','FID3    ',
     #'FID4    ','FID5    ','FID6    ','FID7    ','FID8    '/
      NAMELIST /INPUT/ TIME,TEMP,TOUT,SO,DC,TSTART,TOFF,TOCC,TUNOC,TB
      DATA MS/1/,FUNCTION/Y'00080013'/
      DATA NORESPONSE/'NO RESPONSE FROM FID I/O TASK'/
      DATA IFUNC/2/
  100 READ (3,INPUT,ERR=900,END=9999)
      RETURN
  900 WRITE(5,FMT="(' READ ERROR ON LU #3')")
      GO TO 100
 9999 STOP
      END
C=====================================================================
      SUBROUTINE AHU(COMMAND)
C=====================================================================
      CHARACTER*6 COMMAND
      LOGICAL IWAIT,IPAUSE,ICHANGE
      CHARACTER*8 TNAMES
      COMMON/TASKS/TNAMES(25)
      COMMON/CONTRO/IWAIT,IPAUSE,ICHANGE
      IF(COMMAND.EQ.'START ')THEN
       WRITE(1,FMT="(1X,'*** START AIR HANDLING UNIT ***')")
      ELSE IF(COMMAND.EQ.'STOP  ')THEN
       WRITE(1,FMT="(1X,'*** STOP AIR HANDLING UNIT ***')")
      ELSE
       WRITE(1,FMT="(1X,'*** UNKNOWN AIR HANDLING UNIT COMMAND ***')")
      ENDIF
      RETURN
      END
C=====================================================================
      SUBROUTINE OPSAVE(OPENED,STATUS)
C=====================================================================
C FILE OPENING SUBROUTINE FOR PERKIN ELMER MINICOMPUTER.
C MAY BE SYSTEM DEPENDENT
C
      LOGICAL OPENED
      CHARACTER*6 STATUS
      OPEN(UNIT=2,FILE='OPTSS.SAV ',STATUS=STATUS,RECL=80,SHARE='SRW ',
     &ERR=900)
      OPENED=.TRUE.
      RETURN
  900 OPENED=.FALSE.
      RETURN
      END
```

APPENDIX B. SAMPLE IMPLEMENTATION OF DUTY CYCLING ALGORITHM

```
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE DUTCYC
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C This routine is the main routine to cause duty cycling of electrical
C loads connected to digital outputs to occur. Based on the contents
C of a duty cycle table, the routine causes digital outputs currently
C on to be turned off for certain periods of time to save energy.
C The routine LDCONT is used to create an interface to the loads.
C
C VARIABLE DEFINITIONS:
C
C COMMON BLOCK DUTYCT: The duty cycle table. It is loaded from the CCU.
C LOAD   - array of load ID numbers to be duty cycled (MUX # and point #)
C PCPHAS - array of phase times for the loads (% of duty cycle interval)
C PCOFF  - array of off-period times for the loads(% of duty cycle interval)
C ADJUST - logical array, 1 if off-period is to be dynamically adjusted
C DCAMUX - array of MUX ID numbers for the analog values used for adjustment
C DCAPNT - array of Point ID's for the analog values used for adjustment
C DCADES - array of values for the adjustment analog value at the design point
C DCALO  - array of lower values of the adjustment analog at minimum off-period
C DCAHI  - array of higher values of the adjustment analog at min. off-period

C DCMAST  - task ID number for duty cycler task (passed in common block)
C DCI     - duty cycle interval obtained from task manager task table
C DELTA   - design minus current analog value for off-period adjustment
C FRAC    - difference between design and high (low) adjustment analog values
C H,M,S,TS- task execution interval times from task table (hrs., mins., etc.)
C IEDOS   - argument for task interval utility routine, holds task status
C PHASE   - two part integer containing phase for current load in mins. and sec
C REDUC   - reduction in off-period when duty cycle adjustment is made
C STATUS  - status of request to load controller to turn load off and on
C TIMOFF  - two part integer containing off-period for current load (min.,sec.)
C TPHASE  - absolute phase in seconds for current load
C TTO     - absolute off-period in seconds for current load
C
      INTEGER LOAD,PCPHAS,PCOFF,TTO,TPHASE
      BYTE ADJUST,DCAMUX,DCAPNT
      REAL DCADES,DCALO,DCAHI,DCI
      REAL*8 ANAI
      REAL DELTA,FRAC
      INTEGER REDUC,LOAD,PHASE(2),TIMOFF(2),STATUS
      BYTE CNTRL,SCAN0,SCMAST,SCSLAV,ONOFFT,TSKMAX,DCMAST
      BYTE H,M,S,TS
      COMMON/TSKINT/H,M,S,TS
      COMMON/TSKLNK/CNTRL(2),SCAN0,SCMAST,SCSLAV(16),
     #ONOFFT(16),TSKMAX,DCMAST
      COMMON/DUTYCT/LOAD(16),PCPHAS(16),PCOFF(16),ADJUST(16),DCAMUX(16),
     #DCAPNT(16),DCADES(16),DCALO(16),DCAHI(16)
      COMMON/ANALOG/ANAI(1,32)
```

```
C
      I=DCMAST
      CALL TLOG(I,1,0.)
C************************************
      CALL TLOG(10,1,0.)
C************************************
C------------determine duty cycle interval DCI-----------------------
      CALL TSKCHK(DCMAST,IEDOS)
      DCI=M*60+S
C----------read duty cycle loads and parameters from table-------------
         DO 1000 I=1,16
         IF(LOAD(I).LE.0)GO TO 1000
         RWORK=DCI*PCOFF(I)/100.
         TTO=RWORK
         RWORK=DCI*PCPHAS(I)/100.
         TPHASE=RWORK
C************************************
      R = TTO
      CALL TLOG(10,TPHASE,R)
C************************************
         IF(ADJUST(I).EQ.0)GO TO 500
C----------adjust PCOFF and PCPHAS if required-------------------------
         M=DCAMUX(I)
         S=DCAPNT(I)
         DELTA=DCADES(I)-ANAI(M,S)
         IF(DELTA.LE.0)FRAC=DCADES(I)-DCAHI(I)
         IF(DELTA.GT.0)FRAC=DCADES(I)-DCALO(I)
C************************************
      IN = FRAC
      CALL TLOG(10,IN,DELTA)
C************************************
         IF(FRAC.EQ.0)GO TO 500
         FRAC=DELTA/FRAC
         REDUC=TTO*FRAC
         TPHASE=TPHASE+REDUC
         TTO=TTO-REDUC
C----------------Call load controller to turn load on and off-----------
  500    TIMOFF(1)=TTO/60
         TIMOFF(2)=TTO-TIMOFF(1)*60
         PHASE(1)=TPHASE/60
         PHASE(2)=TPHASE-PHASE(1)*60
C************************************
      R = LOAD(I)
      CALL TLOG(10,4,R)
C************************************
         CALL LDCONT(LOAD(I),PHASE,TIMOFF,5,STATUS)
 1000    CONTINUE
C------------Check status and take appropriate action------------------
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C In this version, no status check is made. Status check becomes       C
C important when demand limit control is added to the FID.             C
```

```
C If records of duty cycling are to be kept, this routine must also    C
C keep track of the actual time that loads are cycled off              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      R=STATUS
      I=DCMAST
      CALL TLOG(I,0,R)
C************************************
      CALL TLOG(10,0,R)
C************************************
      RETURN
      END
C=====================================================================
      SUBROUTINE LDCONT(LOAD,PHASE,TIMOFF,PRIOR,STATUS)
C=====================================================================
C This is the load controller routine. It is used by routines which
C control electrical loads controlled by digital outputs. The routine
C performs the following functions. 1: checks to see if the load is
C currently under control by a routine with a higher priority than
C the routine requesting control; 2: Checks to see if minimum off-time
C criteria are satisfied; 3: Checks for a violation of minimum on-time
C criteria; 4: controls the load by the use of two currently unused
C floating on-off control tasks.
C
C VARIABLE DEFINITIONS:

C CURPRI - current priority level that digital output to be controlled is at
C DOPRI  - array containing current control priority of all digital outputs
C INUSE  - logical array indicating if an on-off task is reserved
C LOAD   - ID number of the load to be cycled off
C MINOFT - minimum value of the off-period to avoid equipment damage
C NEGPRI - negative of PRIOR, used to turn on load and request a priority check
C ONAGIN - relative time interval from present before load is turned back on
C PHASE  - relative time interval from present before load is turned off
C PRIOR  - priority that load is to be controlled under
C STATUS - status of request for load control:
C   status = 0 , loads successfully controlled
C   status = 1 , load control rejected due to low priority request
C   status = 2 , off-time less than minimum specification
C TIMOFF - off-period for load
C
      BYTE DIGO,DOPRI,DIGIN,ON,OFF
      INTEGER LOAD,PHASE(2),TIMOFF(2),PRIOR,STATUS,MINOFT
      INTEGER ONAGIN(2),LOAD2,CURPRI,NEGPRI
      BYTE MUXPNT(2)
      COMMON/DIGITA/DIGO(1,24),DOPRI(1,24),DIGIN(1,16)
      EQUIVALENCE (LOAD2,MUXPNT(1))
      DATA MINOFT/2/,OFF/0/,ON/1/
C-----------------priority check-------------------------------------
      LOAD2=LOAD
      I2=MUXPNT(1)
      I1=MUXPNT(2)
```

```
      IF(I1.NE.0)GO TO 80
      CURPRI = DOPRI(1,23)
      GO TO 90
   80 CURPRI=DOPRI(I1,I2)
   90 IF(PRIOR.LE.CURPRI)GO TO 100
      STATUS=1
      RETURN
C---------minimum on-time check-----------------------------------------
  100 CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C Minimum on-time checking is not implemented in this version. When    C
C demand limit control is added to the FID, this routine must check to C
C see if the load to be turned off has been on long enough to avoid    C
C damage to equipment or satisfy other criteria. If demand limit is    C
C trying to turn off a load, the duty cycler may just have turned a    C
C load on. A minimum on time must elapse, so the time until the load   C
C is turned off, the phase, must be adjusted. If duty cycle is trying  C
C to turn off a load, and demand limit has just released a load, then  C
C there must also be a minimum on time, and the phase must be adjusted.C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C--------minimum off-time check---------------------------------------
C Minimum off time becomes important when the percent off time becomes C
C too small and there is the risk of equipment damage as equipment is  C
C turned off and then on again a short time later. Currently, minimum  C
C is set at 2 MINUTES.                                                 C
C---------------------------------------------------------------------
      IF(TIMOFF(1).GE.MINOFT)GO TO 200
      STATUS=2
      RETURN
C----------set load to turn off and then on-----------------------------
  200 NEGPRI=-PRIOR
C              CALL TO SUBROUTINE DELAYD(MOTSK,OUT,PRIOR,ONOFF,MIN,SEC,NTS)
C*************************************
      R = LOAD
      CALL TLOG(10,5,R)
C*************************************
      CALL DELAYD(2,LOAD,NEGPRI,OFF,PHASE(1),PHASE(2),0)
      ONAGIN(1)=PHASE(1)+TIMOFF(1)
      ONAGIN(2)=PHASE(2)+TIMOFF(2)
      CALL DELAYD(2,LOAD,NEGPRI,ON,ONAGIN(1),ONAGIN(2),0)
      STATUS=0
      RETURN
      END
C=====================================================================
      SUBROUTINE DELAYD(MOTSK,OUT,PRIOR,ONOFF,MIN,SEC,NTS)
C=====================================================================
C this routine is called to turn a load off or on, after a specified time
C interval. An available on-off task is found, where DOTSK is the number
C of the highest on-off task which can be used to control the load. OUT is
C the digital output to control, PRIOR is the priority to control the load
C under, ONOFF is 1 to turn on a load, 0 to turn it off, and MIN and SEC
```

```
C are the time interval that should elapse before the output is controlled.
C COMMON BLOCK ONOFF contains a table of on-off task parameters that are
C set before the task can be used.
C
      INTEGER PRIOR,OUT,MOTSK
      BYTE H,M,S,TS,TSK,ONOFF
      BYTE DONOFF,INUSE
      INTEGER MIN,SEC,IPR,DNUM
      BYTE CNTRL,SCAN0,SCMAST,SCSLAV,ONOFFT,TSKMAX,DCMAST
      COMMON/TSKINT/H,M,S,TS
      COMMON/ONOFF/DNUM(16),DONOFF(16),INUSE(16),IPR(16)
      COMMON/TSKLNK/CNTRL(2),SCAN0,SCMAST,SCSLAV(16),
     #ONOFFT(16),TSKMAX,DCMAST
      DATA NOOTSK/16/
C----------find unused on/off task pair---------------------------------
      ITSK=0
         DO 1000 I=MOTSK,NOOTSK
         IF(INUSE(I).NE.0)GO TO 1000
         ITSK=I
         GO TO 2000
 1000    CONTINUE
      ITSK=16
 2000 INUSE(ITSK)=1
C----------set parameters in on-off task table--------------------------
      INUSE(ITSK)=1
      IPR(ITSK)=PRIOR
      DNUM(ITSK)=OUT
      DONOFF(ITSK)=ONOFF
C----------cause on-off task to execute after a delay time--------------
      TS=NTS
      H=0
      M=MIN
      S=SEC
      CALL TSKEDT(ONOFFT(ITSK),-1,1)
      RETURN
      END
C=============================================================================
      SUBROUTINE TLOG(NTASK,BEGEND,RN)
C=============================================================================
C This routine is used to store information for the task log. The number
C of the task, a code to indicate beginning or end of the task, and a
C number which may be set to a meaningful value by the task, are all
C written to the log buffer.
C
      BYTE MS,TSLOT(32),LF,CR
      INTEGER BEGEND,III
      BYTE ANFLAG,DFLAG,STOPCR,ONDLOG,ONTLOG,INIT
      BYTE TBUFFR,TRESET,TENABL
      COMMON/FLAGS/ANFLAG(1,32),DFLAG(1,24),STOPCR,ONDLOG,ONTLOG,INIT
      COMMON/TLOG/TBUFFR(608),TRESET,NBYTE,TENABL(25)
      COMMON/TIME/MS(9)
```

```
      DATA LF/123/,CR/125/
      IF(ONTLOG.EQ.0)RETURN
      IF(TENABL(NTASK).NE.1.AND.NTASK.GT.0)RETURN
      IF(TRESET.EQ.1)NBYTE=0
      IF(NBYTE.GT.600)GO TO 9000
      TRESET=0
      IF(NBYTE.NE.0)TBUFFR(NBYTE)=LF
      III = BEGEND
      IF(BEGEND.GT.999)III=999
      IF(BEGEND.LT.-99)III=-99
      RRR = RN
      IF(RN.EQ.1.)RRR=0.99999
      CALL RDMSEC
      ENCODE(TSLOT,1)MS(9),MS(8),MS(7),MS(6),MS(5),MS(4),MS(3),MS(2),
     #MS(1),NTASK,III,RRR
    1 FORMAT(2I1,':',2I1,'|',2I1,'.',3I1,' T',I3,I3,1X,G9.3)
      TSLOT(31)=CR
      TSLOT(32)=38
      DO 500 I=1,32
      NBYTE=NBYTE+1
      TBUFFR(NBYTE)=TSLOT(I)
  500 CONTINUE
      RETURN
 9000 TBUFFR(608)=38
      TBUFFR(606)=64
      RETURN
      END
```

APPENDIX C. SAMPLE IMPLEMENTATION OF DEMAND LIMITING ALGORITHM

```
C MODIFIED BY W.B.MAY FOR ACTUAL TESTING OF ALGORITHM ON EMCS
C JUNE 5, 1985
C ***************************************************************
C
C    DLISMAIN  : Demand limiting main program for instantaneous
C                rate method
C
C     January 12, 1984 C.P.
C
C ---------------------------------------------------------------
C
C    DELAY     Delay time to start                          (min)
C    DELP      The amount of power to be shed or restored   (kW)
C    ID        Identification number of a load
C    INITST    True for initial start
C              False when no initial start is needed
C    ITIME     Number of samples taken from the beginning of sampling
C    LDNAME    Load name
C    LOADON    True if the load is turned on
C              False if the load is turned off
C    MAXOFF    Maximum off-time of a load                   (min)
C    MINOFF    Minimum off-time of a load                   (min)
C    MINON     Minimum on-time of a load                    (min)
C    NL        Maximum number of loads (=50)
C    NLD       Total number of loads
C    PDATA     Measured power data                          (kW)
C    PLOAD     Nominal power of a load                      (kW)
C    PMAX      Maximum power allowed in a demand limit period
C                                                           (kW)
C    PMIN      Minimum power allowed in a demand limit period
C                                                           (kW)
C    PRILOW    Lowest global priority
C    PRIOR     Global priority of a load
C              The highest priority is 1 and the lowest priority is
C              PRILOW.
C    PRT       True if printing of detailed information of load status
C              is desired.
C              False if short print-out is desired.
C
C ***************************************************************
C
      LOGICAL   LOADON,INITST,PRT
      REAL      MAXOFF,MINOFF,MINON
      INTEGER   PRIOR,PRILOW
      CHARACTER LDNAME*15
      PARAMETER (NL=50)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,PMIN,PDATA
     &       /BK3/ MAXOFF(NL),MINOFF(NL),MINON(NL),PRIOR(NL),
     &             LDNAME(NL),PLOAD(NL),DELAY(NL),INITST
```

```
     &       /BK4/ NLD,ID(NL),LOADON(NL),PRILOW,LPRIOF(NL,NL),
     &             LPRLOW(NL),LPRION(NL,NL),PRT
      NAMELIST /INPUT/ PMAX,PMIN,PRILOW,PRT,INITST
     &         /OUTPUT/ ITIME,PDATA,DELP
C-----------------------------------------------------------------------
      INTEGER SLICE
      PARAMETER (SLICE = 30)
      EXTERNAL REPLY

      CALL INIT
      CALL ENABLE(2,REPLY)
      CALL CONMSG(23,'INITIALIZATION COMPLETE')
C-----------------------------------------------------------------------
C
C
C     Read input data files and print them.
C
      OPEN(7,FILE='INPUTIS./55')
      OPEN(8,FILE='LOADTABL./55')
C-----OPEN(9,FILE='INPUTPWR')
      REWIND 7
      REWIND 8
C-----REWIND 9
C
      READ(7,INPUT)
      PRINT INPUT
      PRINT 4000
      I=1
10    READ(8,1000,END=20) ID(I),LDNAME(I)
      READ(8,*) ID(I),PRIOR(I),PLOAD(I),DELAY(I),MINOFF(I),
     & MINON(I),MAXOFF(I)
      PRINT 2000, ID(I),LDNAME(I),PRIOR(I),PLOAD(I),DELAY(I),MINOFF(I),
     & MINON(I),MAXOFF(I)
      I=I+1
      GOTO 10
20    NLD=I-1
C
C     Read power signal from a meter in real control
C
      ITIME=0
C30---READ(9,*,END=999) PDATA
C
C-----------------------------------------------------------------------
C READ ACTUAL POWER LEVEL FROM FID
C
 30   CALL DEMAND(5,1,2,PDATA)
C
C-----------------------------------------------------------------------
C
      IF(ITIME.EQ.1441) ITIME=1
      PRINT 3000
```

```
C
C     Instantaneous rate method
C
      CALL DLIS(DELP)
C
C     Control loads based on priorities, minimum on/off-times
C      and maximum on-times of loads.
C
      CALL LDONOF(DELP)
      PRINT OUTPUT
      IF(.NOT.PRT) THEN
        PRINT 5000,(ID(I),I=1,NLD)
        PRINT 6000,(LOADON(I),I=1,NLD)
      ENDIF
      ITIME=ITIME+1
      DELP=0.0
C
C-----------------------------------------------------------------------
C REAL TIME WAIT
C
      CALL WAIT(SLICE,2,IS)
      TSAMPL = FLOAT(SLICE)/60.
C-----------------------------------------------------------------------
C
      GOTO 30
C
1000  FORMAT(I3,1X,A15)
2000  FORMAT(I5,1X,A15,I3,5F10.2)
3000  FORMAT(80('-')/)
4000  FORMAT(//T4,'ID',T9,'ITEM',T21,'PRIORITY',T30,'PLOAD',
     & T40,'DELAY',T50,'MINOFF',T60,'MINON',T70,'MAXOFF'/)
5000  FORMAT(20I4)
6000  FORMAT(20L4)
C
999   STOP
      END
C ***********************************************************************
C
C     DLIS : Demand limiting using instantaneous rate method
C
C     January 12, 1984  C.P.
C
C ----------------------------------------------------------------------
C    DELP     The amount of power to be shed or restored       (kW)
C    PAVG     Average value of current and past powers
C    PDATA    Measured power data                              (kW)
C    PLAG     Time-lagged power                                (kW)
C    PMAX     Maximum power allowed in a demand limit period
C                                                              (kW)
C    PMIN     Minimum power allowed in a demand limit period
C                                                              (kW)
```

```
C    RESET    True when a reset signal is on.
C             False when the reset signal is off.
C ***************************************************************
C
      SUBROUTINE DLIS(DELP)
      LOGICAL   RESET
      DIMENSION PLAG(0:1)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,PMIN,PDATA
      NAMELIST /OUTISW/ PAVG,PMAX,PMIN
      DATA RESET/.TRUE./
C
      IF(RESET) THEN
        PLAG(0)=0.0
        PLAG(1)=0.0
        RESET=.FALSE.
C
      ELSE
        PLAG(0)=PDATA
        PAVG=(PLAG(0)+PLAG(1))/2.
        IF(PAVG.GT.PMAX) THEN
          DELP=PMAX-PAVG
          PRINT 1000,-DELP
        ELSEIF(PAVG.LT.PMIN) THEN
          DELP=PMIN-PAVG
          PRINT 2000,DELP
        ENDIF
C
        PRINT OUTISW
C
        PLAG(1)=PLAG(0)
      ENDIF
C
1000  FORMAT(/5X,'------ POWER TO BE SHED',F10.2,'-----')
2000  FORMAT(/5X,'++++++ POWER TO BE RESTORED',F10.2,'+++++')
C
      RETURN
      END

C MODIFIED BY W.B. MAY FOR ACTUAL TESTING OF ALGORITHM ON EMCS
C JUNE 5, 1985
C ***************************************************************
C
C     DLRFMAIN : Demand limiting main program for the ideal rate method
C                with fixed interval metering
C
C     January 12, 1984 C.P.
C
C ---------------------------------------------------------------
C
C    DELAY    Delay time to start                          (min)
C    DELP     The amount of power to be shed or restored   (kW)
```

```
C    DIFF     Difference between maximum and minimum energy levels
C                                                              (kWh)
C    DMDP     Demand period                                    (min)
C    ID       Identification number of a load
C    INITST   True for initial start
C             False when no initial start is needed
C    IRESET   1 for on-status of the reset signal of demand metering
C             0 for off-status of the reset signal
C    ITIME    Number of samples taken from the beginning of sampling
C    LDNAME   Load name
C    LOADON   True if the load is turned on
C             False if the load is turned off
C    MAXOFF   Maximum off-time of a load                       (min)
C    MINOFF   Minimum off-time of a load                       (min)
C    MINON    Minimum on-time of a load                        (min)
C    NL       Maximum number of loads (=50)
C    NLD      Total number of loads
C    OFFSET   Offset at the beginning of each demand period    (kWh)
C    PDATA    Measured power data                              (kW)
C    PLOAD    Nominal power of a load                          (kW)
C    PMAX     Maximum power allowed in a demand limit period
C                                                              (kW)
C    PRILOW   Lowest global priority
C    PRIOR    Global priority of a load
C             The highest priority is 1 and the lowest priority is
C             PRILOW.
C    PRT      True if printing of detailed information of load status
C             is desired.
C             False if short print-out is desired.
C    RESET    True when a reset signal is on.
C             False when the reset signal is off.
C    TSAMPL   Sampling period                                  (min)
C
C ***************************************************************
C
      LOGICAL   RESET,LOADON,INITST,PRT
      REAL      MAXOFF,MINOFF,MINON
      INTEGER   PRIOR,PRILOW
      CHARACTER LDNAME*15
      PARAMETER (NL=50)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,DIFF,OFFSET,PDATA
     &       /BK2/ RESET,IRESET,ENCAL
     &       /BK3/ MAXOFF(NL),MINOFF(NL),MINON(NL),PRIOR(NL),
     &             LDNAME(NL),PLOAD(NL),DELAY(NL),INITST
     &       /BK4/ NLD,ID(NL),LOADON(NL),PRILOW,LPRIOF(NL,NL),
     &             LPRLOW(NL),LPRION(NL,NL),PRT
      NAMELIST /INPUT/ DMDP,TSAMPL,PMAX,DIFF,OFFSET,PRILOW,PRT,INITST
     &        /OUTPUT/ ITIME,PDATA,IRESET,DELP
C-----------------------------------------------------------------
      INTEGER*2 COUNTS
      INTEGER DIGITIN,DIGITOUT
```

```
      COMMON/DISPLY/DIGITIN(16),DIGITOUT(24),COUNTS(8)
      INTEGER SLICE
      EXTERNAL REPLY

      CALL INIT
      CALL ENABLE(2,REPLY)
      CALL CONMSG(23,'INITIALIZATION COMPLETE')
C-----------------------------------------------------------------------
C
C     Read input data files and print them.
C
      OPEN(7,FILE='INPUTRF./55')
      OPEN(8,FILE='LOADTABL./55')
C-----OPEN(9,FILE='INPUTPWR')
      REWIND 7
      REWIND 8
C-----REWIND 9
C
      READ(7,INPUT)
      PRINT INPUT
      PRINT 4000
C-----------------------------------------------------------------------
      SLICE = TSAMPL * 60
      IRESET = 0
C-----------------------------------------------------------------------
      I=1
10    READ(8,1000,END=20) ID(I),LDNAME(I)
      READ(8,*) ID(I),PRIOR(I),PLOAD(I),DELAY(I),MINOFF(I),
     & MINON(I),MAXOFF(I)
      PRINT 2000, ID(I),LDNAME(I),PRIOR(I),PLOAD(I),DELAY(I),MINOFF(I),
     & MINON(I),MAXOFF(I)
      I=I+1
      GOTO 10
20    NLD=I-1
C
C     Read power and reset signals from a meter in real control
C
      ITIME=0
C30----READ(9,*,END=999) PDATA,IRESET
C-----------------------------------------------------------------------
C READ ACTUAL POWER LEVEL FROM FID
C
  30  CALL DEMAND(5,1,2,PDATA)
      IF(DIGITOUT(16).EQ.1.AND.IRESET.EQ.0)THEN
        IRESET = 1
      ELSE IF(IRESET.EQ.1)THEN
        IRESET = 0
      ENDIF
C-----------------------------------------------------------------------
      IF(ITIME.EQ.0.AND.IRESET.NE.1) GOTO 30
      IF(ITIME.EQ.1441) ITIME=1
```

```
      PRINT 3000
C
C     Ideal rate method using  the fixed interval metering
C
      CALL DLRF(DELP)
C
C     Control loads based on priorities, minimum on/off-times
C      and maximum on-times of loads.
C
      CALL LDONOF(DELP)
      PRINT OUTPUT
      IF(.NOT.PRT) THEN
        PRINT 5000,(ID(I),I=1,NLD)
        PRINT 6000,(LOADON(I),I=1,NLD)
      ENDIF
      ITIME=ITIME+1
      DELP=0.0
C-----------------------------------------------------------------------
C REAL TIME WAIT
C
      CALL WAIT(SLICE,2,IS)
      GOTO 30
C
1000  FORMAT(I3,1X,A15)
2000  FORMAT(I5,1X,A15,I3,5F10.2)
3000  FORMAT(80('-')/)
4000  FORMAT(//T4,'ID',T9,'ITEM',T21,'PRIORITY',T30,'PLOAD',
     & T40,'DELAY',T50,'MINOFF',T60,'MINON',T70,'MAXOFF'/)
5000  FORMAT(20I4)
6000  FORMAT(20L4)
C
999   STOP
      END
C ***********************************************************************
C
C     DLRF : Demand limiting using ideal rate method with
C            fixed interval metering
C
C     January 12, 1984 C.P.
C
C ---------------------------------------------------------------
C    DELP     The amount of power to be shed or restored       (kW)
C    DIFF     Difference between maximum and minimum energy levels
C                                                              (kWh)
C    DMDP     Demand period                                    (min)
C    E        The amount of energy used from the beginning of
C             a demand limit period to the current time        (kWh)
C    EMAX     Maximum energy level allowed  at sampling instant
C                                                              (kWh)
C    EMIN     Minimum energy level allowed at sampling instant
C                                                              (kWh)
```

```
C    ENCAL    Energy used from the beginning of demand period to the
C             sampling instant(i.e., the latest value of E)    (kWh)
C    IRESET   1 for on-status of the reset signal of demand metering
C             0 for off-status of the reset signal
C    N        Number of samples in a demand limit period
C    NINT     Maximum number of samples in a demand limit period(=60)
C    OFFSET   Offset at the beginning of demand period         (kWh)
C    PDATA    Measured power data                              (kW)
C    PMAX     Maximum power allowed in a demand limit period
C                                                             (kW)
C    PWR      Average value of power at current and past sampling
C             instants                                         (kW)
C    RESET    True when a reset signal is on.
C             False when the reset signal is off.
C    TSAMPL   Sampling period                                  (min)
C ***************************************************************
C
      SUBROUTINE DLRF(DELP)
      LOGICAL   RESET
      PARAMETER (NINT=60)
      DIMENSION P(0:NINT),E(0:NINT)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,DIFF,OFFSET,PDATA
     &       /BK2/ RESET,IRESET,ENCAL
      NAMELIST /OUTRFI/ I,PWR,ENCAL,EMAX,EMIN
      DATA ICYCLE,IFLAG/0,1/,E(0)/0.0/
C
C     Bypass  when the reset signal is missed and resume the normal
C     operation when the reset signal appears
C
      IF(IFLAG.EQ.2) THEN
        IF(IRESET.EQ.1) THEN
          IFLAG=1
          ICYCLE=0
        ELSE
          PRINT 3000
          RETURN
        ENDIF
      ENDIF
C
C     Reset the counting of samples
C
      IF(IRESET.EQ.1 .AND. IFLAG.EQ.1) THEN
        N=DMDP/TSAMPL+0.01
        RESET=.TRUE.
        NN=0
      ENDIF
C
C     Set all energy stock values zero at the end of demand period
C     except the initial cycle
C
      IF(RESET) THEN
```

```
        IF(ICYCLE.EQ.0) THEN
          P(0)=PDATA
          ENCAL=0.0
          ICYCLE=1
        ELSE
          P(I)=PDATA
          ENCAL=ENCAL+(P(I)+P(I-1))/2.*TSAMPL/60.
          P(0)=P(I)
        ENDIF
        I=0
        DO 10 K=0,N
10      E(K)=0.0
        RESET=.FALSE.
C
C     Calculate average power and determine the power to be shed
C     or restored.
C
      ELSE
        P(I)=PDATA
        E(I)=(P(I)+P(I-1))/2.*TSAMPL/60.+E(I-1)
        ENCAL=E(I)
        EMAX=(PMAX-60.0*OFFSET/DMDP)*(I*TSAMPL/60.)+OFFSET
        EMIN=EMAX-DIFF
        IF(I.LT.N) THEN
          P(I)=60.*(E(I)-E(I-1))/TSAMPL
          PWR=P(I)
          IF(E(I).GT.EMAX) THEN
            DELP=PMAX-60.*OFFSET/DMDP-P(I)
            PRINT 1000,-DELP
          ELSEIF(E(I).LT.EMIN) THEN
            DELP=PMAX-60.*OFFSET/DMDP-P(I)
            PRINT 2000,DELP
          ENDIF
        ENDIF
      ENDIF
C
      PRINT OUTRFI
      I=I+1
      NN=NN+1
C
C     Set a flag to bypass the calculation when the reset signal misses
C
      IF(NN.GT.N) THEN
        NN=NN-1
        IFLAG=2
        PRINT 3000
      ENDIF
C
1000  FORMAT(/5X,'----- POWER TO BE SHED',F10.2,'-----')
2000  FORMAT(/5X,'+++++ POWER TO BE RESTORED',F10.2,'+++++')
3000  FORMAT('!!!!!!! MISSING RESET SIGNAL  !!!!!'/)
```

```
C
      RETURN
      END

C MODIFIED BY W.B. MAY FOR ACTUAL TESTING OF ALGORITHM ON EMCS
C JUNE 6, 1985
C ***************************************************************
C
C     DLPFSMAIN: Demand limiting main program for predictive method
C
C       January 12, 1984 C.P.
C
C ---------------------------------------------------------------
C
C    DELAY    Delay time to start                              (min)
C    DELP     The amount of power to be shed or restored       (kW)
C    DMDP     Demand period                                    (min)
C    FIXINT   True if fixed interval metering is used
C             False if sliding window metering is used
C    ID       Identification number of a load
C    INITST   True for initial start
C             False when no initial start is needed
C    IRESET   1 for on-status of the reset signal of demand metering
C             0 for off-status of the reset signal
C    ITIME    Number of samples taken from the beginning of sampling
C    LDNAME   Load name
C    LOADON   True if the load is turned on
C             False if the load is turned off
C    MAXOFF   Maximum off-time of a load                       (min)
C    MINOFF   Minimum off-time of a load                       (min)
C    MINON    Minimum on-time of a load                        (min)
C    MODE     = 1 for fixed interval metering
C             = 2 for sliding window metering
C    NL       Maximum number of loads (=50)
C    NLD      Total number of loads
C    PDATA    Measured power data                              (kW)
C    PLOAD    Nominal power of a load                          (kW)
C    PMAX     Maximum power allowed in a demand limit period
C                                                              (kW)
C    PMIN     Minimum power allowed in a demand limit period
C                                                              (kW)
C    PRILOW   Lowest global priority
C    PRIOR    Global priority of a load
C             The highest priority is 1 and the lowest priority is
C             PRILOW.
C    PRT      True if printing of detailed information of load status
C             is desired.
C             False if short print-out is desired.
C    TSAMPL   Sampling period                                  (min)
C
C ***************************************************************
```

```
C
      LOGICAL   RESET,FIXINT,LOADON,INITST,PRT
      REAL      MAXOFF,MINOFF,MINON
      INTEGER   PRIOR,PRILOW
      CHARACTER LDNAME*15
      PARAMETER (NL=50)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,PMIN,PDATA
     &       /BK2/ FIXINT,RESET,IRESET,PPRED,ENCAL
     &       /BK3/ MAXOFF(NL),MINOFF(NL),MINON(NL),PRIOR(NL),
     &             LDNAME(NL),PLOAD(NL),DELAY(NL),INITST
     &       /BK4/ NLD,ID(NL),LOADON(NL),PRILOW,LPRIOF(NL,NL),
     &             LPRLOW(NL),LPRION(NL,NL),PRT
      NAMELIST /INPUT/ DMDP,TSAMPL,PMAX,PMIN,PRILOW,PRT,MODE,INITST
     &         /OUTPUT/ ITIME,PDATA,IRESET,DELP,FIXINT
      DATA FIXINT/.FALSE./
C
C-----------------------------------------------------------------------
      INTEGER*2 COUNTS
      INTEGER DIGITIN,DIGITOUT
      COMMON/DISPLY/DIGITIN(16),DIGITOUT(24),COUNTS(8)
      INTEGER SLICE
      EXTERNAL REPLY

      CALL INIT
      CALL ENABLE(2,REPLY)
      CALL CONMSG(23,'INITIALIZATION COMPLETE')
C-----------------------------------------------------------------------
C
C     Read input data files and print them.
C
      OPEN(7,FILE='INPUTPFS./55')
      OPEN(8,FILE='LOADTABL./55')
C-----OPEN(9,FILE='INPUTPWR')
      REWIND 7
      REWIND 8
C-----REWIND 9
C
      READ(7,INPUT)
      PRINT INPUT
      PRINT 4000
C-----------------------------------------------------------------------
      SLICE = TSAMPL * 60
      IRESET = 0
C-----------------------------------------------------------------------
      I=1
10    READ(8,1000,END=20) ID(I),LDNAME(I)
      READ(8,*) ID(I),PRIOR(I),PLOAD(I),DELAY(I),MINOFF(I),
     & MINON(I),MAXOFF(I)
      PRINT 2000, ID(I),LDNAME(I),PRIOR(I),PLOAD(I),DELAY(I),MINOFF(I),
     & MINON(I),MAXOFF(I)
      I=I+1
```

```
      GOTO 10
20    NLD=I-1
C
C     Read power and reset signals from a meter in real control
C
      ITIME=0
C30----READ(9,*,END=999) PDATA,IRESET
C-------------------------------------------------------------------
C READ ACTUAL POWER LEVEL FROM FID
  30  CALL DEMAND(5,1,2,PDATA)
      IF(DIGITOUT(16).EQ.1.AND.IRESET.EQ.0) THEN
        IRESET = 1
      ELSE IF(IRESET.EQ.1) THEN
        IRESET = 0
      ENDIF
C-------------------------------------------------------------------
      IF(MODE.EQ.1.AND.ITIME.EQ.0.AND.IRESET.NE.1) GOTO 30
      IF(ITIME.EQ.1441) ITIME=1
      PRINT 3000
C
C     Predictive method is called using sliding window or fixed
C     interval metering
C
      CALL DLP(MODE,DELP)
C
C     Control loads based on priorities, minimum on/off-times
C      and maximum on-times of loads.
C
      CALL LDONOF(DELP)
      PRINT OUTPUT
      IF(.NOT.PRT) THEN
        PRINT 5000,(ID(I),I=1,NLD)
        PRINT 6000,(LOADON(I),I=1,NLD)
      ENDIF
      ITIME=ITIME+1
      DELP=0.0
C-------------------------------------------------------------------
C REAL TIME WAIT
      CALL WAIT(SLICE,2,IS)
C-------------------------------------------------------------------
      GOTO 30
C
1000  FORMAT(I3,1X,A15)
2000  FORMAT(I5,1X,A15,I3,5F10.2)
3000  FORMAT(80('-')/)
4000  FORMAT(//T4,'ID',T9,'ITEM',T21,'PRIORITY',T30,'PLOAD',
     & T40,'DELAY',T50,'MINOFF',T60,'MINON',T70,'MAXOFF'/)
5000  FORMAT(20I4)
6000  FORMAT(20L4)
C
999   STOP
```

```
      END
C ***********************************************************************
C
C     DLP  : Demand limiting link program for predictive method
C
C     January 12, 1984 C.P.
C
C ***********************************************************************
C
      SUBROUTINE DLP(MODE,DELP)
      LOGICAL   RESET,FIXINT
      COMMON /BK1/ DMDP,TSAMPL,PMAX,PMIN,PDATA
     &       /BK2/ FIXINT,RESET,IRESET,PPRED,ENCAL
C
C     Sliding window metering
C
      IF(MODE.EQ.2) THEN
        CALL DLPSB(DELP)
C
C     Fixed interval metering
C     If the reset signal is missing at the next sampling
C     instance, the sliding window metering is activated. When
C     the reset signal is restored, the fixed interval metering
C     is also restored.
C
      ELSEIF(MODE.EQ.1) THEN
        IF(IRESET.EQ.1) THEN
          FIXINT=.TRUE.
        ENDIF
        IF(FIXINT) THEN
          CALL DLPFB(DELP)
        ELSE
          CALL DLPSB(DELP)
        ENDIF
      ENDIF
C
      RETURN
      END
C ***********************************************************************
C
C     DLPFB: Demand limiting using predictive method with
C            fixed interval metering
C
C     January 12, 1984 C.P.
C
C ---------------------------------------------------------------------
C    DELP     The amount of power to be shed or restored       (kW)
C    DMDP     Demand period                                    (min)
C    E        The amount of energy used from the beginning of
C             a demand limit period to the current time        (kWh)
C    EMAX     Maximum energy level allowed in a demand limit period
```

```
C                                                                    (kWh)
C    EMIN     Minimum energy level allowed in a demand limit period
C                                                                    (kWh)
C    ENCAL    Energy used during a sampling period                   (kWh)
C    EPRED    Predicted value of energy use during a demand period
C                                                                    (kWh)
C    EPS      A small positive number (=0.01)
C    FIXINT   True if the fixed interval method is used
C             False if the sliding window method is used
C    IRESET   1 for on-status of the reset signal of demand metering
C             0 for off-status of the reset signal
C    N        Number of samples in a demand limit period
C    NINT     Maximum number of samples in a demand limit period(=60)
C    P        Power at a sampling instant                            (kW)
C    PDATA    Measured power data                                    (kW)
C    PMAX     Maximum power allowed in a demand limit period
C                                                                    (kW)
C    PMIN     Minimum power allowed in a demand limit period
C                                                                    (kW)
C    PPRED    Predicted value of average power for a demand limit
C             period                                                 (kW)
C    RESET    True when a reset signal is on.
C             False when the reset signal is off.
C    TSAMPL   Sampling period                                        (min)
C ***********************************************************************
C
      SUBROUTINE DLPFB(DELP)
      LOGICAL   RESET,FIXINT
      PARAMETER (NINT=60)
      DIMENSION P(0:NINT),E(0:NINT)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,PMIN,PDATA
     &       /BK2/ FIXINT,RESET,IRESET,PPRED,ENCAL
      NAMELIST /OUTPFI/ I,PPRED,EPRED,EMAX,EMIN,ENCAL
      DATA ICYCLE/0/
C
C     Reset the counting of samples
C
      IF(IRESET.EQ.1) THEN
        N=DMDP/TSAMPL+0.01
        RESET=.TRUE.
        NN=0
      ENDIF
C
C     Set all energy stock values zero at the end of demand period
C     except the initial cycle
C
      IF(RESET) THEN
        IF(ICYCLE.EQ.0) THEN
          P(0)=PDATA
          ENCAL=0.0
          ICYCLE=1
```

```
        ELSE
          P(I)=PDATA
          ENCAL=ENCAL+(P(I)+P(I-1))/2.*TSAMPL/60.
          P(0)=P(I)
        ENDIF
        I=0
        DO 10 K=0,N
10      E(K)=0.0
        RESET=.FALSE.
C
C     Predict energy use at the end of demand period and
C     determine the power to be shed or restored.
C
      ELSE
        P(I)=PDATA
        E(I)=(P(I)+P(I-1))/2.*TSAMPL/60.+E(I-1)
        ENCAL=E(I)
        EMAX=PMAX*DMDP/60.
        EMIN=PMIN*DMDP/60.
        IF(I.LT.N) THEN
          EPRED=(N-I)*(E(I)-E(I-1))+E(I)
          PPRED=60.*EPRED/DMDP
          IF(EPRED.GT.EMAX) THEN
            DELP=60.*(EMAX-EPRED)/(DMDP-I*TSAMPL)
            PRINT 1000,-DELP
          ELSEIF(EPRED.LT.EMIN) THEN
            DELP=60.*(EMIN-EPRED)/(DMDP-I*TSAMPL)
            PRINT 2000,DELP
          ENDIF
        ENDIF
      ENDIF
      PRINT OUTPFI
      I=I+1
      NN=NN+1
C
C     Switch over to the sliding window metering when the
C     reset signal is not detected.
C
      IF(NN.GT.N) THEN
        ICYCLE=0
        FIXINT=.FALSE.
        RESET=.TRUE.
        PRINT 3000
      ENDIF
C
1000  FORMAT(/5X,'----- POWER TO BE SHED',F10.2,'-----')
2000  FORMAT(/5X,'+++++ POWER TO BE RESTORED',F10.2,'+++++')
3000  FORMAT('!!!!!!! SWITCHED TO SLIDING WINDOW !!!!!'/)
C
      RETURN
      END
```

```
C ***************************************************************
C
C     DLPSB : Demand limiting using predictive method with
C             sliding window metering
C
C     January 12, 1984 C.P.
C
C ---------------------------------------------------------------
C    DELP     The amount of power to be shed or restored       (kW)
C    DMDP     Demand period                                    (min)
C    E        The amount of energy used from the beginning of
C             a demand limit period to the current time        (kWh)
C    EMAX     Maximum energy level allowed in a demand limit period
C                                                              (kWh)
C    EMIN     Minimum energy level allowed in a demand limit period
C                                                              (kWh)
C    ENCAL    Energy used during a sampling period             (kWh)
C    EPRED    Predicted value of energy use during a demand period
C                                                              (kWh)
C    N        Number of samples in a demand limit period
C    NINT     Maximum number of samples in a demand limit period(=60)
C    PDATA    Measured power data                              (kW)
C    PLAG     Time-lagged power                                (kW)
C    PMAX     Maximum power allowed in a demand limit period
C                                                              (kW)
C    PMIN     Minimum power allowed in a demand limit period
C                                                              (kW)
C    PPRED    Predicted value of average power for a demand limit
C             period                                           (kW)
C    RESET    True when a reset signal is on.
C             False when the reset signal is off.
C    TSAMPL   Sampling period                                  (min)
C ***************************************************************
C
      SUBROUTINE DLPSB(DELP)
      LOGICAL   RESET
      PARAMETER (NINT=60)
      DIMENSION PLAG(0:NINT),E(0:2)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,PMIN,PDATA
     &       /BK2/ FIXINT,RESET,IRESET,PPRED,ENCAL
      NAMELIST /OUTPSW/ I,PPRED,EPRED,EMAX,EMIN,ENCAL
C
C     Determine the number of samples per demand period
C
      N=DMDP/TSAMPL+0.01
C
C     Initialize the regressor vector of power.
C
      IF(RESET) THEN
        ICYCLE=0
        I=0
```

```
        DO 10 K=0,N
10      PLAG(K)=0.0
        RESET=.FALSE.
C
C     Determine the regressor vector of power in the learning period.
C
      ELSEIF(ICYCLE.EQ.0) THEN
        PLAG(0)=PDATA
        DO 20 K=N-1,0,-1
20      PLAG(K+1)=PLAG(K)
        IF(I.EQ.N-1) ICYCLE=1
        I=I+1
C
C     Predict energy use at the next sampling instance
C     and determine the power to be shed or restored.
C
      ELSE
        PLAG(0)=PDATA
        SUM=0.0
        DO 30 K=1,N-1
30      SUM=SUM+TSAMPL/60.*(PLAG(K)+PLAG(K+1))/2.
        E(0)=SUM
        E(1)=TSAMPL/60.*(PLAG(0)+PLAG(1))/2.+E(0)
        ENCAL=E(1)
        E(2)=2*E(1)-E(0)
        EPRED=E(2)
        PPRED=60.*EPRED/DMDP
        EMAX=PMAX*DMDP/60.
        EMIN=PMIN*DMDP/60.
        IF(EPRED.GT.EMAX) THEN
          DELP=60.*(EMAX-EPRED)/TSAMPL
          PRINT 1000,-DELP
        ELSEIF(EPRED.LT.EMIN) THEN
          DELP=60.*(EMIN-EPRED)/TSAMPL
          PRINT 2000,DELP
        ENDIF
C
        PRINT OUTPSW
C
C     Shift back the  regressor vector by one sample period
C
        DO 40 K=N-1,0,-1
40      PLAG(K+1)=PLAG(K)
      ENDIF
C
1000  FORMAT(/5X,'------ POWER TO BE SHED',F10.2,'-----')
2000  FORMAT(/5X,'++++++ POWER TO BE RESTORED',F10.2,'+++++')
C
      RETURN
      END
```

```
C MODIFIED JUNE 6, 1985 BY W.B. MAY TO CONTROL FID POINTS.
C ***********************************************************************
C
C     LDONOF :  Turn on or off loads
C
C     April 29, 1983 C.P.
C
C -------------------------------------------------------------------
C
C    DELAY    Delay time to start                                  (min)
C    DELP     The amount of power to be shed or restored           (kW)
C    EPS      Small positive number for tolerance(=0.01)
C    INITST   True for initial start
C             False when no initial start is needed
C    ISTAT    Number of loads which are turned on at an initial stage
C    ISUM     Number of loads turned on during initial cycle
C    LOADON   True if the load is turned on
C             False if the load is turned off
C    LPR      Local priority of a load
C    LPRIOF   Local priority of a load for shedding
C    LPRION   Local priority of a load for restoring
C    LPRLOW   Lowest local priority level
C    MAXOFF   Maximum off-time of a load                           (min)
C    MINOFF   Minimum off-time of a load                           (min)
C    MINON    Minimum on-time of a load                            (min)
C    NL       Maximum number of loads (=50)
C    NLD      Total number of loads
C    PLOAD    Nominal power of a load                              (kW)
C    PRI      Global priority of a load
C    PRILOW   Lowest global priority
C    PRIOR    Global priority of a load
C             The highest priority is 1 and the lowest priority is
C             PRILOW.
C    PRT      True if printing of detailed information of load status
C             is desired.
C             False if short print-out is desired.
C    SUMPINT  Summed nominal power turned on during initial cycle
C    SUMP     Summed nominal power actually shed or restored at an instant
C                                                                   (kW)
C    TOFF     Off-time of a load                                   (min)
C    TON      On-time of a load                                    (min)
C    TSAMPL   Sampling period                                      (min)
C
C ***********************************************************************
C
      SUBROUTINE LDONOF(DELP)
C
      REAL      MAXOFF,MINOFF,MINON
      INTEGER   PRIOR,PRI,PRILOW
      LOGICAL   LOADON,INITST,PRT
      CHARACTER LDNAME*15
```

```
      PARAMETER (NL=50)
      DIMENSION TON(NL),TOFF(NL),ISTAT(NL)
      COMMON /BK1/ DMDP,TSAMPL,PMAX,PMIN,PDATA
      COMMON /BK3/ MAXOFF(NL),MINOFF(NL),MINON(NL),PRIOR(NL),
     &             LDNAME(NL),PLOAD(NL),DELAY(NL),INITST
     &       /BK4/ NLD,ID(NL),LOADON(NL),PRILOW,LPRIOF(NL,NL),
     &             LPRLOW(NL),LPRION(NL,NL),PRT
      NAMELIST /NAM1/ SUMINT,ISUM,INITST
     &         /NAM2/ SUMP
      DATA EPS/0.01/,IFLAG/0/
C-----------------------------------------------------------------------
      LOGICAL SHED(NL)
      INTEGER*2 COUNTS
      INTEGER DIGITIN,DIGITOUT
      COMMON/DISPLY/DIGITIN(16),DIGITOUT(24),COUNTS(8)
C-----------------------------------------------------------------------
C
C     Initialization
C
      IF(IFLAG.EQ.0) THEN
        DO 10 I=1,NLD
        TON(I)=0.0
        TOFF(I)=0.0
C------ LOADON(I)=.FALSE.
C-----------------------------------------------------------------------
        SHED(I) = .FALSE.
C-----------------------------------------------------------------------
        ISTAT(I)=0
10      CONTINUE
C
C     Set up local priority levels for each global priority
C     level, PRIOR(I).  Two local priority levels are assigned
C     in a sequential order, one for turn-off and another for
C     turn-on.
C
        DO 40 PRI=1,PRILOW
        K=0
        DO 20 I=1,NLD
        IF(PRIOR(I).EQ.PRI) THEN
          K=K+1
          LPRIOF(PRI,I)=K
        ENDIF
20      CONTINUE
        LPRLOW(PRI)=K
        KK=LPRLOW(PRI)+1
        DO 30 I=1,NLD
        IF(PRIOR(I).EQ.PRI) THEN
          KK=KK-1
          LPRION(PRI,I)=KK
        ENDIF
30      CONTINUE
```

```
40        CONTINUE
          IFLAG=1
        ENDIF
C
C-----------------------------------------------------------------------
C make loads depend on status inputs
          DO 15 I = 1,NLD
          IF(DIGITIN(I).EQ.0) LOADON(I) = .TRUE.
          IF(DIGITIN(I).EQ.1) LOADON(I) = .FALSE.
  15      CONTINUE
C-----------------------------------------------------------------------
C
C
C       Turn on loads during the initial cycle after delay times
C       are over.
c
C
        IF(INITST) THEN
          ISUM=0
          SUMINT=0.0
          DO 50 I=1,NLD
          IF(TOFF(I).GE.DELAY(I)) THEN
C---------LOADON(I)=.TRUE.
C
C-----------------------------------------------------------------------
C control load in FID
          CALL DIGOUT(5,1,I+16,.TRUE.,-4)

C-----------------------------------------------------------------------
C
            SUMINT=SUMINT+PLOAD(I)
            ISTAT(I)=1
          ENDIF
          ISUM=ISUM+ISTAT(I)
          IF(ISUM.EQ.NLD) INITST=.FALSE.
50        CONTINUE
        PRINT NAM1
        ENDIF
C
C       Shed loads if power decrease is demanded by the
C       amount of DELP, and if minimum on-times are passed.
C       Start to shed loads with low priority first.
C       Assign the highest local priority to the load most
C       recently shed.
C
        IF(DELP.LT.-EPS) THEN
          SUMP=0.0
          DO 90 PRI=PRILOW,1,-1
60        DO 80 LPR=LPRLOW(PRI),1,-1
          DO 70 I=1,NLD
          IF(PRIOR(I).GE.PRI.AND.TON(I).GE.MINON(I).AND.MAXOFF(I).GT.
```

```
     &    0.0.AND.LOADON(I).AND.LPRIOF(PRI,I).EQ.LPR) THEN
C---------LOADON(I)=.FALSE.
C
C-----------------------------------------------------------------------
C control load in FID
        CALL DIGOUT(5,1,I+16,.FALSE.,-4)
        SHED(I) = .TRUE.
C-----------------------------------------------------------------------
C
          TOFF(I)=0.0
          KEY=LPRIOF(PRI,I)
          CALL PUSH(KEY,PRI)
          SUMP=SUMP+PLOAD(I)
          PRINT 1000,I
          GOTO 60
        ENDIF
        IF(SUMP.GE.-DELP) GOTO 100
70      CONTINUE
80      CONTINUE
90      CONTINUE
100     PRINT NAM2
      ENDIF
C
C     Restore loads if power increase is allowed by the
C     amount of DELP, and if minimum off-times are passed.
C     Start to restore loads with high priority first.
C     Assign the lowest local priority to the load most
C     recently restored.
C
      IF(DELP.GT.EPS) THEN
        SUMP=0.0
        DO 140 PRI=1,PRILOW
110     DO 130 LPR=1,LPRLOW(PRI)
        DO 120 I=1,NLD
        IF(PRIOR(I).EQ.PRI.AND.TOFF(I).GE.MINOFF(I).AND.
     &    (.NOT.LOADON(I)).AND.LPRION(PRI,I).EQ.LPR.AND.SHED(I)) THEN
C---------LOADON(I)=.TRUE.
C
C-----------------------------------------------------------------------
C control load in FID
        CALL DIGOUT(5,1,I+16,.TRUE.,-4)
        SHED(I) = .FALSE.
C-----------------------------------------------------------------------
C
          TON(I)=0.0
          ISTAT(I)=1
          KEY=LPRION(PRI,I)
          CALL POP(KEY,PRI)
          SUMP=SUMP+PLOAD(I)
          PRINT 2000,I
          GOTO 110
```

```
        ENDIF
        IF(SUMP.GE.DELP) GOTO 150
120     CONTINUE
130     CONTINUE
140     CONTINUE
150     PRINT NAM2
      ENDIF
C
C     Restore loads regardless of priority level, if maximum
C     off-times are passed.
C
      IF(.NOT.INITST) THEN
        DO 160 PRI=1,PRILOW
        DO 160 I=1,NLD
        IF(PRIOR(I).EQ.PRI.AND.TOFF(I).GE.MAXOFF(I).AND.SHED(I)
     &    .AND.(.NOT.LOADON(I))) THEN
C---------LOADON(I)=.TRUE.
C
C-----------------------------------------------------------------------
C control load in FID
        CALL DIGOUT(5,1,I+16,.TRUE.,-4)
        SHED(I) = .FALSE.
C-----------------------------------------------------------------------
C
          TON(I)=0.0
          KEY=LPRION(PRI,I)
          CALL POP(KEY,PRI)
          PRINT 5000,I
        ENDIF
160      CONTINUE
      ENDIF
C
C     Print details
C
      IF(PRT) THEN
        PRINT 3000
        DO 170 I=1,NLD
170     PRINT 4000,I,PLOAD(I),TOFF(I),MINOFF(I),TON(I),MINON(I),
     &  LOADON(I),PRIOR(I),LPRIOF(PRIOR(I),I),LPRION(PRIOR(I),I)
      ENDIF
C
C     Increase on- and off-times by one sample period for
C     the use in the next time step
C
      DO 180 I=1,NLD
      IF(LOADON(I)) THEN
        TON(I)=TON(I)+TSAMPL
        IF(TON(I).GE.2*MINON(I)+24*60) TON(I)=2*MINON(I)
      ELSE
        TOFF(I)=TOFF(I)+TSAMPL
        IF(TOFF(I).GE.2*MAXOFF(I)+24*60) TOFF(I)=2*MAXOFF(I)
```

```
      ENDIF
180   CONTINUE
C
1000  FORMAT(5X,'------ LOAD #',I5,2X,'SHED-----')
2000  FORMAT(5X,'++++++ LOAD #',I5,2X,'RESTORED++++++')
3000  FORMAT(/T5,'I',T10,'PLOAD',T21,'TOFF',T29,'MINOFF',T41,'TON',
     &T50,'MINON',T56,'STATUS',T63,'PRI',T67,'LPRIOF',T74,'LPRION'/)
4000  FORMAT(I5,5(F9.2,1X),3X,L1,2X,I4,2(3X,I3,1X))
5000  FORMAT(5X,'++++++ LOAD #',I5,2X,'RESTORED SINCE ',
     & 'THE MAXIMUM OFF-TIME IS PASSED')
C
      RETURN
      END
C ***********************************************************************
C
C     PUSH :  Determine local priority levels for each given
C               global priority .
C
C             Assign the highest local priority to the load
C             which is most recently shed.
C
C     April 27, 1983 C.P.
C
C ***********************************************************************
C
      SUBROUTINE PUSH(KEY,PRI)
      LOGICAL   LOADON,PRT
      INTEGER   PRI,PRILOW
      PARAMETER (NL=50)
      COMMON /BK4/ NLD,ID(NL),LOADON(NL),PRILOW,LPRIOF(NL,NL),
     &             LPRLOW(NL),LPRION(NL,NL),PRT
      DIMENSION ITEMP(NL,NL)
C
      DO 10 I=1,NLD
        IF(LPRIOF(PRI,I).LT.KEY) THEN
          ITEMP(PRI,I)=LPRIOF(PRI,I)+1
        ELSEIF(LPRIOF(PRI,I).GT.KEY) THEN
          ITEMP(PRI,I)=LPRIOF(PRI,I)
        ELSE
          ITEMP(PRI,I)=1
        ENDIF
10    CONTINUE
C
      DO 20 I=1,NLD
20    LPRIOF(PRI,I)=ITEMP(PRI,I)
C
      RETURN
      END
C ***********************************************************************
C
C     POP :  Determine local priority levels for each given
```

```
C               global priority .
C
C            Assign the lowest local priority to the load
C            which is most recently restored.
C
C
C     April 27, 1983 C.P.
C
C ***************************************************************************
C
      SUBROUTINE POP(KEY,PRI)
      LOGICAL   LOADON,PRT
      INTEGER   PRI,PRILOW
      PARAMETER (NL=50)
      COMMON /BK4/ NLD,ID(NL),LOADON(NL),PRILOW,LPRIOF(NL,NL),
     &             LPRLOW(NL),LPRION(NL,NL),PRT
      DIMENSION ITEMP(NL,NL)
C
      DO 10 I=1,NLD
        IF(LPRION(PRI,I).GT.KEY) THEN
          ITEMP(PRI,I)=LPRION(PRI,I)-1
        ELSEIF(LPRION(PRI,I).LT.KEY) THEN
          ITEMP(PRI,I)=LPRION(PRI,I)
        ELSE
          ITEMP(PRI,I)=LPRLOW(PRI)
        ENDIF
10    CONTINUE
C
      DO 20 I=1,NLD
20    LPRION(PRI,I)=ITEMP(PRI,I)
C
      RETURN
      END
```

APPENDIX D. SAMPLE IMPLEMENTATION OF DEMAND SUPPLY AIR RESET ALGORITHM

```
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE DRESET
C::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C This routine is used to reset the supply air setpoint of an air
C handling unit. This version uses the demand determination method, where
C the demand is determined from the measured usage of reheat coils in the
C zones served by the air handler. The setpoint is moved up or down by a
C step change when the reheat usage is outside of specified limits.
C Common block SETPNT contains the set points used by the AHU task,
C which regulates the supply air.
C
C VARIABLE DEFINITIONS:
C
C DRTSKI - task interval(s) at which DRESET executes. From FID task table.
C DRSAMP - sampling period (s)for adjusting setpoint and summing reheat.
C RHMAX  - setpoint raised if percentage of total samples with reheat on
C          exceeds this value.
C RHMIN  - setpoint lowered if percentage of total samples with reheat on
C          is less than this value.
C RHSTEP -
C
      INTEGER ARSTAT
      INTEGER RHZONE(2)
      BYTE DIGO,DOPRI,DIGIN
      REAL TREF
      BYTE H,M,S,TS
C
      COMMON/TSKINT/H,M,S,TS
      COMMON/DIGITA/DIGO(1,24),DOPRI(1,24),DIGIN(1,16)
      COMMON/RHDEMA/RHMAX(5),RHMIN(5),RHSTEP(5),DRSAMP
      COMMON/AIRSET/TOAMAX(5),TOAMIN(5),TRMIN(5),TRMAX(5)
     #,TOAMID(5),TRMID(5),ARSTAT(5)
      COMMON/SETPNT/TREF(5)
      DATA RHZONE/2*0/,MCOUNT/0/,NRHZON/1/
C------------determine execution interval ------------------------
      CALL LOCK
      CALL TSKCHK(7,IEDOS)
      DRTSKI=M*60+S
      CALL TLOG(7,1,DRTSKI)
      MAXCNT=DRSAMP/DRTSKI
C--------determine reheat coil usage-----------------------------
      DO 100 I=1,NRHZON
      J=I+8
      RHZONE(I)=RHZONE(I)+1-DIGIN(1,J)
  100 CONTINUE
      MCOUNT=MCOUNT+1
      IF(MCOUNT.LT.MAXCNT)GO TO 9000
```

```
      MCOUNT=0
C------Determine minimum reheat usage for all zones--------------
      IDZONE=0
      MINRHV=MAXCNT
      DO 200 I=1,NRHZON
      IF(RHZONE(I).GT.MINRHV)GO TO 190
      IDZONE=I
      MINRHV=RHZONE(I)
  190 RHZONE(I)=0
  200 CONTINUE
      RHPCT=FLOAT(MINRHV)/FLOAT(MAXCNT)*100.
C--------change setpoint value as a function of reheat usage-------
C*****************************************************************
C       CALL TLOG(7,IDZONE,RHPCT)
C*****************************************************************
      DO 5000 I=1,5
      IF(ARSTAT(I).NE.2)GO TO 5000
C--------if air temperature is outside of limits, fix setpoint----------
      IF(TREF(I).GT.TRMAX(I))TREF(I)=TRMAX(I)
      IF(TREF(I).LT.TRMIN(I))TREF(I)=TRMIN(I)
      IF(RHPCT.GT.RHMAX(I))GO TO 3000
      IF(RHPCT.LE.RHMIN(I))GO TO 4000
C--------no change in setpoint desired----------------------------------
      GO TO 5000
C--------too much reheat - raise setpoint-------------------------------
 3000 ERROR=RHPCT-RHMAX(I)
      TREF(I)=TREF(I)+RHSTEP(I)*ERROR
      IF(TREF(I).GT.TRMAX(I))TREF(I)=TRMAX(I)
      GO TO 5000
C--------not enough reheat - lower setpoint ----------------------------
 4000 ERROR=RHPCT-RHMIN(I)
      TREF(I)=TREF(I)+RHSTEP(I)*ERROR
      IF(TREF(I).LT.TRMIN(I))TREF(I)=TRMIN(I)
 5000 CONTINUE
 9000 CONTINUE
      CALL TLOG(7,0,TREF(1))
      CALL UNLOCK
      RETURN
      END
C===============================================================================
      SUBROUTINE TLOG(NTASK,BEGEND,RN)
C===============================================================================
C This routine is used to store information for the task log. The number
C of the task, a code to indicate beginning or end of the task, and a
C number which may be set to a meaningful value by the task, are all
C written to the log buffer.
C
      BYTE MS,TSLOT(32),LF,CR
      INTEGER BEGEND
      BYTE ANFLAG,DFLAG,STOPCR,ONDLOG,ONTLOG,INIT
      BYTE TBUFFR,TRESET,TENABL
```

```
      COMMON/FLAGS/ANFLAG(1,32),DFLAG(1,24),STOPCR,ONDLOG,ONTLOG,INIT
      COMMON/TLOG/TBUFFR(608),TRESET,NBYTE,TENABL(25)
      COMMON/TIME/MS(9)
      DATA LF/123/,CR/125/
      IF(ONTLOG.EQ.0)RETURN
      IF(TENABL(NTASK).NE.1.AND.NTASK.GT.0)RETURN
      IF(TRESET.EQ.1)NBYTE=0
      IF(NBYTE.GT.600)GO TO 9000
      TRESET=0
      IF(NBYTE.NE.0)TBUFFR(NBYTE)=LF
      CALL RDMSEC
      ENCODE(TSLOT,1)MS(9),MS(8),MS(7),MS(6),MS(5),MS(4),MS(3),MS(2),
     #MS(1),NTASK,BEGEND,RN
    1 FORMAT(2I1,':',2I1,'|',2I1,'.',3I1,' T',I3,I3,1X,G9.3)
      TSLOT(31)=CR
      TSLOT(32)=38
      DO 500 I=1,32
      NBYTE=NBYTE+1
      TBUFFR(NBYTE)=TSLOT(I)
  500 CONTINUE
      RETURN
 9000 TBUFFR(608)=38
      TBUFFR(606)=64
      RETURN
      END
```

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 85-3285	2. Performing Organ. Report No.	3. Publication Date JANUARY 1986

4. TITLE AND SUBTITLE

VERIFICATION OF PUBLIC DOMAIN CONTROL ALGORITHMS FOR BUILDING ENERGY MANAGEMENT AND CONTROL SYSTEMS

5. AUTHOR(S)

William B. May, Jr. and George E. Kelly

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)

U.S. Department of Energy
1000 Independence Avenue, SW
Washington, DC 20585

U.S. Naval Civil Engineering Laboratory
Port Hueneme, CA 93043

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)

Software is an important component of building energy management and control systems (EMCS). It is usually supplied by EMCS vendors in proprietary packages that do not contain human readable source code. Even when source code listings are made available on a "limited use" basis, it is difficult for the user to determine whether the supplied algorithms meet the design specifications because of the lack of public domain HVAC control algorithms with which to compare them.

To help overcome the above problem, the National Bureau of Standards developed and documented eight public domain EMCS supervisory control algorithms. The testing and verification of these eight algorithms are described in this report. The algorithms tested cover dry bulb and enthalpy economizer cycles, optimum and scheduled start/stop, duty cycling, demand limiting, outside air supply air reset, and demand supply air reset. For each of these algorithms, the process of installing the algorithms on an NBS laboratory system is discussed and a description is given of the tests performed. The results of these experimental studies are presented, along with any additional considerations for use of the algorithms that were developed as a result of the testing program.

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

algorithm verification; control algorithms; demand limiting; duty cycling; economizer cycles; energy management algorithms; field testing; HVAC control; optimum start/stop; temperature reset

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

141

15. Price

$16.95

USCOMM-DC 6043-P80