A COMPARISON OF FIXED FLOW AND VARIABLE FLOW RATE CONTROLLERS FOR FLAT PLATE WATER COOLED SOLAR COLLECTORS

by

Douglas R. Bryant

March, 1982

A Master of Science thesis submitted to the
Department of Mechanical Engineering

University of Manitoba

A COMPARISON OF FIXED FLOW AND VARIABLE FLOW RATE CONTROLLERS FOR FLAT PLATE WATER COOLED SOLAR COLLECTORS

BY

DOUGLAS R. BRYANT

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ACKNOWLEDGEMENT

The author would like to thank Mr. Abe Abdelkerim for his encouragement and support and to Mr. Egbert Verbrugge of Multiple Access Limited for allowing the author the use of his company's computer system for plotting the data. A special thanks to Professor Ray Chant for his support and patience.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF FIGURES - continued

NOMENCLATURE

The following list of symbols are used in the text. A prime (') may accompany any variable as a superscript to indicate the value of the variable after a time step. Also the subscripts i and j may accompany any variable to indicate that the variable is calculated from i to j. A single subscript of i or j indicates the variable is a property only of node i or node j.

A	cross sectional area (ft^2)
C	thermal capacity (Btu/°F)
C_p	specific heat at constant pressure (Btu/lbm°F)
F	radiation form factor (dimensionless)
G	convection, conduction, or mass flow conductor (Btu/hr°F)
$\hat{G}$	radiation conductor (Btu/hr°R^4)
h_c	unit convective conductance (Btu/hrft^2°F)
k	thermal conductivity (Btu/hrft°F)
L	length (ft)
$\dot{m}$	fluid mass flow rate (lbm/hr)

Q	heat source or sink (Btu/hr)
S	heat source or sink (°F/hr)
t	time (hrs)
T	temperature (°F)
T_{fi}	inlet fluid temperature (°F)
T_{fo}	outlet fluid temperature (°F)
V	volume (ft^3)
α	thermal diffusivity (ft^2/hr)
β	factor relating forward and backward differencing (dimensionless)
Δ^2	$\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$
Δ	indicates a difference for the parameter it is associated with
ε	emittance (dimensionless)
ρ	density (lbm/ft^3)
σ	Stefan-Boltzmann constant (0.1712×10^{-8} $Btu/hrft^2°R^4$)

ABSTRACT

The purpose of this investigation is to compare the performance of fixed flow and variable flow rate controllers on a flat plate water cooled solar energy collector system using computer simulation techniques. System performance for each type of controller is achieved by constructing a thermal math model of the solar collector, storage system, and controller, and then using a finite differencing computer program to calculate the temperatures in the system, and daily system efficiencies. The boundary conditions required to drive the computer program were taken from climatic data collected on the Manitoba Legislative Building Solar Demonstration Project, and the Winnipeg International Airport.

A total of 7 days, with widely varying conditions, were analyzed. Also, five maximum flow rates, covering a large range, were set for each type of controller giving a total of 70 simulations. The results of these simulations consistently indicate that the amount of energy collected by variable flow rate controllers offer significant increases in performance only at very high flow rates. Both controllers have virtually the same daily efficiencies when operated at the lower, more practical flow rates usually used.

1.0 INTRODUCTION

The high costs of flat plate solar energy collector systems has led to a requirement to reduce the cost per unit of collected energy. Many research projects are examining methods of improving system performance by finding ways of increasing the net energy collected by the absorber plate. This work concentrates on one of these methods with the aim of increasing the flat plate solar energy collector system's economic viability.

Reducing heat loss from a flat plate collector is one method of increasing collector performance and, perhaps, (depending on how costly it is to obtain the increased performance) increasing the energy per unit cost. Suppressing convection between the ambient and the absorber plate by using such methods as honeycomb structures, evacuated cells, multiple cover plates, etc. have been considered as ways of reducing heat losses. Another method is to reduce radiative heat losses by coating the glass cover plates on the side facing the absorber plate with a material which is transparent to solar radiation but reflects infrared radiation. All of these methods usually add significant costs to a flat plate collector or are self defeating in that the amount of solar energy eventually reaching the absorber plate is significantly reduced. Still another method can be used to reduce heat losses to the ambient. The amount of heat lost is dependent upon the thermal resistance from the collector plate to the ambient

and the temperature difference between these two points. Most of the methods mentioned previously deal with increasing this thermal resistance. The method investigated in this work deals with reducing the temperature difference between the collector plate and the ambient.

Collector plate temperature is determined primarily by three heat sources or sinks

(a) the amount of solar energy,
(b) the amount of heat lost to the environment,
(c) the amount of heat transferred to the coolant.

The first is a heat source and is controlled by local weather conditions, while the second, a heat sink, may be adjusted by the methods outlined in the previous paragraph. The third is also a heat sink and may be controlled by varying the flow rate of coolant through the collector, thus controlling collector plate temperature. A low flow rate would give a higher plate to ambient temperature difference maximizing heat lost to the ambient. Conversely, a very high flow rate would maintain a small plate to ambient temperature difference thus minimizing heat losses. Unfortunately, high flow rates require larger pipes and valving, consume more power, and present more of an erosion corrosion problem than do low flow rates. Clearly, the optimum flow rate lies somewhere in between. This work compares the performance gains, on flat plate collectors, obtained by different flow rates.

Collector plate temperature fluctuations can adversely affect the coolant controller thus reducing the amount of energy collected. Solar energy daily profiles are basically sinusoidal with a maximum occurring at 12:00 hours solar time. Cloud coverage often causes steep, lengthy drops in solar energy availability. The effects of these non-uniform solar energy profiles causes large collector plate temperature changes. These temperature fluctuations can adversely affect the performance of solar energy systems which use fixed flow rate controllers (i.e., the flow is either zero or full flow - no intermediate flow rates). Fixed flow rate controllers usually incorporate a deadband, similar to home thermostats which prevents rapid cycling of the pump. However, after the pump has shut off, the collector temperature must rise to some preset value above the shut off value before the pump is activated. Therefore, energy is lost because the pump remains idle when it may be able to collect energy. A pump controller, capable of maintaining constant collector plate temperatures should be more efficient than one that allows collector plate temperatures to fluctuate. A variable flow rate controller can maintain, within certain limits, a much more constant collector plate temperature by varying the coolant flow rate according to the difference between the storage tank and the collector plate temperatures. This study examines these two controllers to determine if the variable flow rate controller is more efficient than the fixed flow rate controller.

Summarizing then, the purpose of this work is not to optimize a particular controller or to simulate an actual solar energy system. The objective is to compare two types of controllers operating under identical conditions using realistic climatic data (climatic data are a major factor in determining controller performance). By limiting the scope of this work to a comparison, it is not necessary to include all of the components or attributes of a real solar energy system since the effects of such components or attributes cancel when comparing the two controllers (as long as the components or attributes are the same for both controllers). Therefore variation of system parameters such as pumping power, inclusion of tank stratification, coolant fluid properties, storage tank capacity, collector angles various controller deadbands, etc. were not examined because it was not necessary and because parametric studies would greatly exceed the scope and time limitations of this work. For example the computer time required to examine, say, four different controller deadbands would have exceeded eight weeks of continuous 24 hours a day running. Inclusion of varying other system parameters would have multiplied the effort by several times. The author believes that the 70 simulations presented in this work are adequate to indicate whether or not the generally more expensive variable flow rate controller is superior to the on/off controller.

2.0 SOLUTION TECHNIQUE

This section reviews previous work, examines the various methods available to predict solar energy system performance, and concludes by outlining the technique used in this thesis.

2.1 General

The best method of comparing the performance of the two types of controllers is to construct two identical solar energy collection systems. One system would operate using a fixed flow controller while the other would use a variable flow controller. Both systems would be closely monitored and operated simultaneously under identical conditions. Operation over a wide variety of environments would allow an accurate comparison of each controller's performance. Unfortunately this approach was beyond the resoures available. The problem, then, was to predict solar energy performance, using the two types of controllers, without relying upon solar energy collector hardware.

2.2 Literature Search

An extensive literature search revealed that little work has been carried out in determining the effect of flow rate and controller type upon solar energy collector system performance. Kovarik and Lesse(1) developed an analytical model describing collector performance as a function of

mass flow rate. Because of the complexity of the equations, this model does not include local forcing functions (i.e., insolation, wind speed, etc.) and it deals only with an on/off flow rate controller, not a variable flow rate controller. However, the conclusions do indicate that determining the optimal flow rate is a problem most suited for solution by numerical techniques.

The vast majority of the literature available on solar systems deal with experimental or actual intalled systems. The author believes this is probably because the cost to do a proper analysis used to exceed the cost of actually building a system. However, with the decreasing costs of computing power, certain numerical techniques are becoming economically attractive (discussed in Section 4.2). A literature search concerning solar energy system modelling revealed that several researchers have done work related to simulating solar collector performance without using solar collector hardware. Siebers and Viskanta(2), Grossman, et al.(3), and Prabkhar et al.(4) developed analytical models of a solar collector. Unfortunately these analyses are only in two dimensions and they do not use real boundary conditions, again, because of the complexity of the equations. Arafa et al(6) indicate the need for a multi-dimensional collector model by concluding that for transient analysis it is inadequate to treat the solar collector as a single node as is often done in analytical studies. Thomas and Vaughan(6) point out that real boundary conditions must be used if the performance of a solar collector is to be

determined with any accuracy. Duffie and Beckman(7) illustrate that solar system performance is best predicted using numerical techniques.

Although these previous works indicate that numerical techniques are the best method, it is advisable to review the advantages and disadvantages of the various methods available for simulating solar energy system performance in order to gain a proper perspective.

2.3 Review of Analytical and Numerical Methods

A mathematical model of a solar system can be developed by mathematically describing the solar energy collector system(7). In developing such models, many simplifying assumptions must be made, particularly when describing the boundary conditions, in order to make the equations solvable. Also, solar collector plates are often treated as large isothermal plates in the steady-state or as having uniform thermal conductances to the environment(7). However, the collector temperatures and conductances vary spatially as well as temporally. Also, real boundary conditions, such as solar insolation, ambient temperature, wind speed and direction, are difficult to describe analytically. Radiation complicates the problem by making the heat transfer and energy balance equations non-linear. The cumulative effect of attempting to exactly model the solar energy collector system and its boundary conditions leads to extremely complex solutions.

Alternatives to exact solutions are approximate numerical techniques. Approximation methods

involve reducing the real system to a network of components that simulate the real system. Each discrete component or node corresponds to an element in the real system and has all of the elements' thermal attributes such as capacitance, conductance, radiation properties, etc. This is often termed a lumped node representation of the system(8). The lumped nodes are interconnected by thermal resistances (conduction, convection, radiation and mass flow) to form a network. Errors are introduced by assuming that the nodes' properties are uniform throughout the node. However, the error can be made small by decreasing the node size (thus increasing the number of nodes). Obviously, a large network of very small nodes would represent the real system with great accuracy; however, practical considerations are often the major factor in determining node size.

Using the lumped node approach, two distinct methods are available to predict solar energy system performance.

(1) The resulting network of a lumped node representation looks very much like an electrical network(8) if electrical symbols are substituted for thermal resistance, capacitances, heat sources and sinks, etc. An analog circuit board could be constructed using capacitors, resistors, diodes, voltages, currents, etc., to represent node thermal capacitance, inter-node thermal conductivity, inter-node mass flow, boundary temperatures, heat sources, etc. By measuring voltages at various points in the

network, equivalent solar energy collection system temperatures could be deduced. However, this system would be costly to construct (there may be hundreds of components which would have to be hand assembled) and difficult to change. If, for example, the collector plate material were changed or the conductances were calculated incorrectly it would be very difficult and time consuming to install the correct values after the system had been wired together. Also, there is the problem of simulating complex boundary conditions and radiation heat transfer, for which there is no known simple electrical equivalent. The analog circuit board approach is possible but would have drawbacks.

(2) Another method of solving a lumped node network is to describe the network mathematically and solve the resulting set of equations. Two common methods of mathematically describing the system are finite elements and finite differences.

Finite elements(9),(10) are used primarily in structural, fluid flow, and thermally induced stress analysis. The finite element procedure is an approximation applied to the variational form of the general heat equation. Generally, the problem reduces to finding a function which minimizes the integral of a partial differential equation with respect to each of it's parameters over the field of interest. When applied to a network of nodes, a system of simultaneous

equations result. The solutions to these equations give the temperatures at various points in the system. Although finite element methods are quite suitable for this work, a simpler approach is available.

Finite differences (see References (8), (9), (10), (11), (12) and (13) are conceptually and mathematically simpler than finite elements and are commonly employed in thermal analysis where stress is not an important factor. The method used in finite differences is to replace the derivative in the general heat equation, which is written for each node in the network, by an algebraic approximation. The resulting set of equations are then solved for each of the unknown temperatures. Knowing the temperature in the solar energy collection system, the system's performance can be deduced. This is the method that was chosen to predict the system's performance and is discussed in greater detail in Section 3.

Solving the set of equations produced by the finite difference method is an iterative procedure (for the implicit case discussed in Section 3.2) which results in N equations and N unknowns where N is the number of nodes. Clearly, a large number of equations will result from a thermal math model which accurately describes the real system by using a large number of nodes. The solution of these equations must be carried out for each time step in a transient problem. Thus, if there are M time steps there will be M x N equations to solve iteratively. The only practical way of solving this set of equations is by using a digital computer.

2.4 An Outline of the Method Used

A computer program has been written in Fortran(14) and Assembler(15) languages to solve the set of equations produced by the finite difference approximation method. Fortran was used for most of the program while Assembler was used to speed up certain sections. The program was developed for, and run on, the author's microcomputer. This machine was assembled by the author from a number of individual components specifically for the purpose of solving sets of equations at high speed. It's principal hardware features are a Zilog Z-80A(16) microprocessor, 64K of memory, an Intel 8232(17) floating point processor, and over 2 megabytes of on-line storage. Special software was written for the floating point processor to provide much faster arithmetic (four basic functions plus square root, cube root, logarithms, etc) than is available through the standard Fortran library. The resulting computer program is well documented and easy to understand. Section 4 provides detailed information on both the computer and the software.

After the computer program was written and tested, the physical system was chosen and then modelled or reduced to a network solvable by finite differencing. Most solar energy collection systems are flat plate water cooled solar collectors connected to a water storage tank(18). The type of collector chosen for this study is the PPG Standard Aluminum Collector. The author became familiar with this collector from his work on the Manitoba Legislative Building, Solar

Demonstration Project(19),(20), and believes it to be typical of flat plate collectors that can be used in the Canadian environment. This collector consists of a flat aluminum plate, painted black, integral tubing, and two glass cover plates. The water storage tank capacity was set at a size typical for short term energy storage. The water flow rates in the model were either variable or fixed flow rate, depending on the controller being tested. Boundary conditions were taken from the Winnipeg International Airport Meteorological Station(21),(22) and data collected from the Manitoba Legislative Building Solar Demonstration Project(20). Although it was intended to make these simulation runs as realistic as possible, the purpose of this thesis is to compare the performance of two control systems, not to determine precise actual performance. Thus details such as exact tank sizing become unimportant as long as the size is in the correct range. After thermally modelling the system (discussed in more detail in Section 5), the resulting 124 nodes (consisting of 124 heat sources/sinks, and 124 capacitances), 413 conductors (consisting of conduction, convection, radiation and mass flow conductors), and the boundary conditions, were submitted to the computer program for evaluation.

A total of 70 simulations were run; 10 for each of the 7 days analyzed. The 7 days represented wide climatic variations thus giving reasonable confidence in the consistency of the results. The 10 runs for each day consisted of 5 simulations for each type of controller. The 5 simulations

each imposed a maximum possible flow rate for each controller allowing a comparison between controllers over a wide range. Each simulation produced about 20,000 temperatures giving a total of about 1.4 million temperatures for all runs. From these temperatures, system performance was determined by comparing energy collected to energy available. The results were 7 graphs showing system efficiency as a function of flow rate and controller type. Since the volume of data were so large, it would be impractical to include all of the supporting data for these 7 graphs. Therefore, a summary of four graphs for typical simulations are included. The four graphs consist of:

(a) flow rate vs. time,

(b) instantaneous energy vs. time,

(c) efficiency vs. (average plate temperature - ambient temperature)/solar insolation, and

(d) temperature vs. time.

The results of the simulation runs are discussed in Section 6.

3.0 FINITE DIFFERENCES

This section discusses the concept of finite differencing and its specific applications to solving heat transfer problems. The heat equation is presented and reduced to a form suitable for describing nodes in a lumped parameter network. The solution of the set of equations, written about each node, are shown to be possible by iterative techniques on a computer. The representation of a physical system's attributes in a lumped parameter network is also presented.

3.1 General

A physical system can be thought of as being composed of an infinite number of infinitely small components. Since the components are infinitely small, this assemblage behaves in exactly the same manner as the real system. Enlarging the elements slightly, connecting each element to its neighbour with a thermal conductance, and assuming that their properties (e.g., conductivity, emittance, etc.) remain constant throughout the component, the assemblage will react almost the same as the real system. Obviously an error has been introduced by discretizing the physical system, but the error can usually be made acceptable by selecting small enough elements. When a network or mesh is used to approximate a thermal system, it is called a thermal math model. This is the concept upon which finite differences rely: the discretization of a homogeneous system.

For each component or node that represents a

portion of the real system, an equation can be written describing the node's energy balance. This equation, when written in its exact form, is the familiar heat equation or the general diffusion equation which describes temperature spatially and temporally. The thermal behaviour of the system is described by this partial differential equation. The resulting set of partial differential equations can be finite differenced such that the set of partial differential equations is reduced to a set of algebraic equations which can then be solved by relaxation techniques. The result of solving the equations is a new value for the temperature of each node. These temperatures correspond directly to discrete locations in the physical system. Temperature profiles or isotherms can then be drawn approximating the temperature at locations between nodes.

The errors inherent in this method can be reduced to acceptable levels by increasing the number of nodes. In areas where great accuracy is required, the node size is made small. However, increasing the number of nodes to gain accuracy increases the modelling effort and increases the computing time (more equations must be solved). Choosing the optimum node size is usually based upon the factors mentioned above and previous experience(11).

3.2 Finite Differencing

Once the physical system has been reduced to a collection of nodes connected to each other by

thermal conductances, the heat equation can be used to describe the node's temperature, spatially and temporally. But, since the physical system is no longer considered a continuum, then the analytical version of the heat equation cannot be used. The heat equation must be discretized or put in its finite difference form to be valid in describing the nodal network.

The heat equation is of the form

$$\partial T/\partial t = \alpha \Delta^2 T + S \qquad (1)$$

Assume that material properties are constant and that heat transfer is linear, that is, radiation heat transfer is not present. (Variable material properties and radiation heat transfer are treated later). The left side of equation (1) can be finite differenced temporally as:

$$\partial T/\partial t = (T' - T)/\Delta t \qquad (2)$$

where the prime indicates the new value after the Δt time period.

Substituting (2) into the left side of (1), it can be seen that the new or predicted temperature is based only on the temperature found in the previous time step.

This type of finite differencing is known as forward differencing. Forward differencing produces a set of explicit equations, when written for the entire nodal network, from which the new temperature can be calculated. Unfortunately this

method invokes a restriction on the time step if it is to remain stable(11).

Allowing the material properties to vary with time and re-writing equation (1) as:

$$(T' - T)/\Delta t = \alpha'\nabla^2 T' + S' \qquad (3)$$

gives a backward differenced equation because the set of equations produced when (3) is written for the entire node network is implicit in T'. The set of equations must be solved simultaneously because there exists, in each equation, more than one unknown. This method does not restrict the time step and is unconditionally stable. However, the solution of the equations usually takes much longer because they must be solved simultaneously (usually by relaxation methods)(12).

A third method of finite differencing is known as the Crank-Nicholson method. This method uses a half forward and a half backward difference. Equation (1) becomes:

$$(T' - T)/\Delta t = \tfrac{1}{2}(\alpha\nabla^2 T + S) + \tfrac{1}{2}(\alpha'\nabla^2 T' + S') \qquad (4)$$

This set of equations, when written for each node in the network is unconditionally stable. However, it is also implicit in T'. Therefore, a solution of simultaneous equations is required. Generally, solution by the Crank-Nicholson method is faster(11) than backward differencing when solving the equations by relaxation.

A more general form of (1) can be written as:

$$\partial T/\partial t = \beta(\alpha\nabla^2 T + S) + (1 - \beta)(\alpha'\nabla^2 T' + S') \qquad (5)$$

$$\text{for } 0 \leq \beta \leq 1.$$

If $\beta = 1$, then forward differencing (explicit equations and stability criteria) results. When $1/2 < \beta < 1$ then an implicit set of equations with stability criteria exist. For $\beta = 1/2$ then forward-backward differencing or the Crank-Nicholson equations (implicit equations, no stability restrictions) are produced. If $0 \leq \beta \leq 1/2$, then backward differencing (implicit equations, no stability restrictions) result[10]. The Crank-Nicholson method is preferred for this work because it gives the fastest solution with an unconditionally stable set of equations.

Let $\beta = 1/2$ and, using the x-direction as an example, let

$$\partial T/\partial x^+ = (T^+ - T)/\Delta x^+$$

$$\partial T/\partial x^- = (T - T^-)/\Delta x^-$$

where the negative and positive signs indicate the direction relative to the node. Using this notation and differencing (5) gives:

$$\alpha\nabla^2 T + S \simeq \frac{\alpha}{\Delta x}\left(\frac{\partial T}{\partial x^+} - \frac{\partial T}{\partial x^-}\right) + \frac{\alpha}{\Delta y}\left(\frac{\partial T}{\partial y^+} - \frac{\partial T}{\partial y^-}\right) +$$

$$\frac{\alpha}{\Delta z}\left(\frac{\partial T}{\partial z^{+}} - \frac{\partial T}{\partial z^{-}}\right) + S \qquad (6)$$

$$\alpha'\nabla^2 T' + S' \simeq \frac{\alpha}{\Delta x}\left(\frac{\partial T'}{\partial x^{+}} - \frac{\partial T'}{\partial x^{-}}\right) + \frac{\alpha'}{\Delta y}\left(\frac{\partial T'}{\partial y^{+}} - \frac{\partial T'}{\partial y^{-}}\right) +$$

$$\frac{\alpha'}{\Delta z}\left(\frac{\partial T'}{\partial z^{+}} - \frac{\partial T'}{\partial z^{-}}\right) + S' \qquad (7)$$

Using Figure 1 as a guide, the x-direction portion of equation (6) can be written as:

$$\frac{\alpha}{\Delta x}\left(\frac{\partial T_o}{\partial x^{+}} - \frac{\partial T_o}{\partial x^{-}}\right) \simeq \frac{\alpha}{\Delta x}\left(\frac{T_2 - T_o}{\Delta x^{+}} - \frac{T_o - T_1}{\Delta x^{-}}\right) \qquad (8)$$

which also applies to the remaining axes of (6) and (7). Letting the node volume $V = (\Delta X)(\Delta Y)(\Delta Z)$, substituting $\alpha = k/\rho C_p$, and simplifying gives:

$$\frac{\rho V C_p}{\Delta t}(T_o' - T_o) = \tfrac{1}{2}\left\{\frac{kAx}{\Delta x^{-}}(T_1 - T_o) + \frac{kAx}{\Delta x^{+}}(T_2 - T_o) + \right.$$

$$\frac{kAy}{\Delta y^{-}}(T_3 - T_o) + \frac{kAy}{\Delta y^{+}}(T_4 - T_o) + \frac{kAz}{\Delta z^{-}}(T_5 - T_o) +$$

$$\left.\frac{kAz}{\Delta z^{+}}(T_6 - T_o) + Q_o\right\} + \tfrac{1}{2}\left\{\frac{k'Ax}{\Delta x^{-}}(T_1' - T_o') + \right. \qquad (9)$$

$$\frac{k'Ax}{\Delta x^{+}}(T_2' - T_o') + \frac{k'Ay}{\Delta y^{-}}(T_3' - T_o') + \frac{k'Ay}{\Delta y^{+}}(T_4' - T_o') +$$

$$\left.\frac{k'Az}{\Delta z^{-}}(T_5' - T_o') + \frac{k'Az}{\Delta z^{+}}(T_6' - T_o') + Q_o'\right\}$$

where $Ax = \Delta y . \Delta z$, $Ay = \Delta x . \Delta z$, $Az = \Delta x . \Delta y$ and $Q = \rho V C_p S$. The term $kAx/\Delta x$ is simply the conductance from node 1 to node 0. Simplifying (9), generalizing for N nodes connected to the i^{th} node, and assuming Q_i is constant over the time step (valid for small time steps) gives:

$$\frac{2C_i}{\Delta t}(T_i' - T_i) = 2Q_i + \sum_{j=1}^{N} G_{ji}(T_j' - T_i') + \sum_{j=1}^{N} G_{ji}(T_j' - T_i')$$

$$1 \leq i \leq N \qquad (10)$$

The product of the conductance and the temperature difference between node i and j is simply the net amount of energy transferred between node i and j. Therefore, by analogy, radiation heat transfer can be included in (11) as:

$$\frac{2C_i}{\Delta t}(T_i' - T_i) = 2Q_i + \sum_{j=1}^{N} \{G_{ji}(T_j - T_i) + \hat{G}_{ji}(T_j^4 - T_i^4)\} +$$

$$\sum_{j=1}^{N} \{G_{ji}(T_j' - T_i') + \hat{G}_{ji}(T_j'^4 - T_i'^4)\} \qquad (11)$$

$$1 \leq i \leq N$$

The only unknowns in equation (11) are T_i' and T_j'. If an iterative or relaxation procedure(12) is used to solve the sets of equations describing the nodes in the network, then the value used for T_j' is the guessed value or the most recent value available from calculations done on node j.

Equation (11) must now be put in a form such that T_i' may be calculated, assuming T_j' is known.

Rearranging (11) gives:

$$(T_i')^4 + c\,T_i' + d = 0 \qquad 1 \le i \le N \tag{12}$$

where

$$c = \frac{\frac{2C_i}{\Delta t} + \sum_{j=1}^{N} G_{ji}}{\sum_{j=1}^{N} \hat{G}_{ji}} \tag{13}$$

and

$$d = -\left\{\frac{2C_iT_i}{\Delta t} + 2Q_i + \sum_{j=1}^{N} G_{ji}(T_j - T_i) + \sum_{j=1}^{N} \hat{G}_{ji}(T_j^4 - T_i^4) + \sum_{j=1}^{N} G_{ji}T_j' + \sum_{j=1}^{N} \hat{G}_{ji}T_j'^4\right\} \Big/ \sum_{j=1}^{N} \hat{G}_{ji} \tag{14}$$

$$1 \le i \le N$$

Equation (12) is a special quartic equation. References (23) and (24) give the solution to this type of quartic. A solution to (12) is as follows:

Equation (12) has the resolvent cubic equation

$$y^3 - 4dy - c^2 = 0 \tag{15}$$

The roots of this equation can be used (by the methods of reference (24)) to obtain the roots of (12). The roots of (15) are given by reference (23) as:

Let

$$p = 4d/3 \tag{16}$$

and

$$q = c^2/2 \tag{17}$$

then for q^2-p^3 positive, (note that d is negative, therefore p^3 is negative), there is one real root:

$$y_1 = \sqrt[3]{(q+(q^2-p^3)^{\frac{1}{2}})} + \sqrt[3]{(q-(q^2-p^3)^{\frac{1}{2}})} \tag{18}$$

The other two roots are imaginary. Now let

$$R = \sqrt{y_1} \tag{19}$$

then

$$E = \sqrt{-y_1 + 2c/\sqrt{y_1}} \tag{20}$$

giving the only, meaningful, real root for (12) as

$$T_i' = -R/2 + E/2 \tag{21}$$

The solution of (12) for each node in the network provides a new value of the temperature for the node. After several iterations through the node network, the new temperatures converge for each time step.

The accuracy of the solution is limited only by the accuracy of the constants, round-off error in the calculations and the amount of computation time available. For most practical situations, computation is halted when the calculation of T_i' from the most recent iteration differs from the calculation of T_i' from the previous iteration by

some small value. The value used in this work was 0.05 °F (0.03 °C).

Also, the time step used in the program was varied between 30 seconds and two minutes. Short time steps were used when the flow rate was high in order to prevent more heat from being withdrawn from a node by the water than the node could supply. Larger time steps were used for low or zero flow rates to reduce computation time.

Equation (12) gives the new temperatures for diffusion nodes (nodes with capacitance) only. Boundary nodes have no capacitance; their new values are calculated by interpolating known data.

3.3 Representation of Physical Parameters

The equations developed in Section 3.2 contain parameters which describe the thermal characteristics of the nodal network. These parameters can be broken down into two basic categories: nodal and inter-nodal.

Nodal parameters consist of thermal capacitances and heat sources or sinks. The thermal capacitance (Btu/°F, (J/°C)), C_i in equation (12), is simply a product of the node's mass and it's specific capacitance. The heat sources or sinks, Q_i in equation (12), are found by summing all heat sources or sinks (Btu/hr, (W)) impressed on the node. Note that both C_i and Q_i can change with respect to time, however, they are assumed constant during the time step increment because, in this work, the time steps are very small. That is, C_i and Q_i cannot change while the set of equations is being solved.

Inter-nodal parameters consist only of conductors. Conductors breakdown into four types: 1) conductivity, 2) convection, 3) radiation, and 4) mass flow. Conductivity can be found by using the following formula:

$$G_{ji} = \frac{k_{ji} A_{ji}}{L_{ji}} \qquad (22)$$

Convective conductance is found from:

$$G_{ji} = h_{cji} A_i \qquad (23)$$

Radiation conductance is calculated by:

$$\hat{G}_{ji} = \sigma \varepsilon_j A_j F_{ji} \qquad (24)$$

The first three types of conductances allow heat transfer in both directions. Mass flow conductances, such as an incompressible fluid flowing from one node to another, allow heat transfer only in the direction of fluid flow.

$$G_{ji} = \dot{m}_{ji} C_p \qquad (25)$$

Until now, only diffusion nodes (i.e., nodes which can store heat) have been discussed. Another type of node, called a boundary node, differs from the diffusion node in that it cannot store heat. Boundary nodes are connected to the diffusion nodes and are used to drive the thermal network or model by applying transient temperature fluctuations at certain locations on the model. In this work, boundary node temperatures are found by interpolating known ambient data.

4.0 THE COMPUTER PROGRAM

This section discusses the type of computer system the thermal analyzer program was run on and then highlights the features of the program.

4.1 General

In Section 3, the equations used to solve for the temperatures in a thermal math model were derived. Since the equations each contain more than one unknown, they must be solved iteratively by using a relaxation procedure. Each equation requires many arithmetic operations such as square roots, cube roots, squares, cubes, as well as multiplications, divisions, additions and subtractions. Because of the large number of calculations required, a computer must be used to solve for the temperatures in the thermal math model. Section 4.2 briefly discusses the type of computer used in this work and indicates how the software was written to take advantage of the computer's hardware.

The software written to solve the set of equations was designed to give flexibility as well as speed. It is written in both Fortran(14) and Assembler(15) languages. Because all of the software is very well documented, only a brief description of programs and subroutines are presented in Section 4.3. It is highly recommended that the reader refer to the software listings in Appendices I, II and III which contain complete descriptions of each software unit's function, inputs, outputs, subroutines referenced

and restrictions as to its use.

4.2 The Computer System

The computer used in this work was a Z-80 microcomputer. It's hardware features are: a Z-80A microprocessor([16]), 64 Kbytes of random access memory, 2 megabytes of disk storage via two double sided, double density, 8-inch floppy disk drives, a clock, a floating point processor (FPP)(17),(25) and several I/O ports. The software used on this machine is the CP/M(26) operating system with the Cromemco Fortran IV)(14) and Cromemco Z-80 Macro Assembler(15).

Although this machine and the software represented a state-of-the-art microcomputer system, the program developed in this thesis taxed the system's capabilities. Originally the floating point processor hardware was not available and the program was written entirely in Fortran. However, it was found, during the development phase, that program execution times were excessive (about 24 hours per simulation). Obviously the development of the program, which involved a large number of runs before it was debugged, required increased speed to run the 70 simulations.

Two bottlenecks in the program were found. The first bottleneck was found in the subroutine called SEARCH. This subroutine is used extensively in subroutine SOLVE. Rewriting SEARCH in assembly language (see Appendix II) significantly reduced execution times. The second bottleneck was due to the way Fortran does

arithmetic. Most microcomputers perform all arithmetic in software, that is, the CPU does multiplication by executing a multiply subprogram. An alternative to this approach is to use special hardware which is designed specifically to do high speed arithmetic(17),(25). The Intel 8232 (a faster version of the Advanced Micro Devices 9512 mentioned in Appendix III) FPP is one such special purpose integrated circuit. When the FPP was installed and the software was written (see Appendix III) to interface it to the thermal analyzer program, further reductions in execution times resulted. The increase in speed was due not only to the faster computations provided by the FPP, but also because the Z-80A and the FPP were set up for pipelining or parallel processing which allowed simultaneous execution of the software. The results of these improvements reduced execution times by about a factor of 15.

For comparison purposes, the author ran the thermal analyzer program, using a Fortran version of subroutine SEARCH, on a computer service bureau's CDC 6600 computer system. The cost for a single run was approximately $6000. Therefore, for 70 simulations, the cost would have been approximately $420,000. The microcomputer costs about $10,000. Based on these costs it is easy to justify the purchase of the microcomputer for this kind of work.

Anyone considering running the thermal analyzer program on a service bureau computer system should bear these costs in mind. Translating the code

given in Appendix II into the service bureau's assembly code would probably be well worth the effort. It is doubtful that implementing the FPP routines would offer any speed improvement over the service bureau computer system. However, if the thermal analyzer were to be run on a microprocessor the user would be well advised to install a floating point processor and implement the software given in Appendices II and III.

4.3 The Thermal Analyzer Program

The thermal analyzer program was written in both Fortran and assembler language. The main program and most of the subroutines are written in Fortran. One subroutine is written in assembler to provide a faster search for adjoining nodes than is possible using Fortran. A number of other subroutines were written in assembler to replace the Fortran supplied arithmetic routines and use the Intel 8232 floating point processor for arithmetic computations. It must be emphasized that this section presents only a brief overview of the thermal analyzer program. Much more comprehensive information may be found in the listings of Appendices I, II and III. This is in accordance with good programming practice which requires, among other things, that the software be self-contained; the software should contain it's own documentation. All of the software presented in the Appendices was written by the author with the exception of subroutine DCSINT which was taken from the Association for Computing Machinery Algorithm Number 547(27).

The main program listed in Appendix I, is called THERMAL. It calls upon many subroutines, most of which are in the Fortran libraries, to calculate the new temperatures in the thermal math model. The first task of THERMAL is to read in the run parameters from file RUNDATA.DAT. This file contains the information necessary to do several simulation runs. The next step is to load all of the parameters that describe the physical system, from file CLCTRDAT.DAT, and (e.g., from file B78001.DAT) and then load the boundary conditions. This is accomplished by subroutine GETINP. The input data are then converted to the proper units via subroutine CONVRS and the boundary data are set up for spline interpolation using subroutine DCSINT. Node areas are converted to square feet and then the radiation conductors are set by subroutine STRADK. Initial temperatures are set to the ambient temperature to give the first approximation or guess in the iterative solution of the set of equations written for each node. This is a good first guess since, after the collector has cooled overnight, the temperature of the entire collector would be very close to the ambient temperature. Note that all of the simulations start prior to sunrise. Finally subroutine SOLVE is called to solve the set of equations and predict the new temperatures for each node.

Subroutine PRE is called by subroutine SOLVE (the listing for SOLVE is in the thermal library in Appendix I) prior to calculating the new temperatures for each time step. This subroutine contains all the logic necessary to simulate a

fixed flow and a variable flow rate solar energy controller. Fluid flow conductances are set by subroutines STFLOW and STTUBK. The time step is adjusted in this routine to ensure that solution accuracy is maintained. Boundary conditions are recalculated by interpolating the boundary data using function SPLEVL and set by routines STPLHC and STFLUX.

Subroutine POST is called by SOLVE after the set of equations has been solved. This allows adjustment of values prior to entering subroutines PRE or PRNOUT. It is used in this program to restore the correct heat flux values in the Q array using functions SPLEVL and subroutine STFLUX.

Subroutine PRNOUT is called by SOLVE at preset intervals. Its prime function to allow printing the current temperatures, capacitances, conductances, etc. This routine also prints the current time, the time between printouts, the total number of iterations and a few other parameters concerning the program's progress.

Subroutine SEARCH is an assembly language routine that has been designed to replace a slower Fortran routine. The function of this subroutine is to perform a binary search through the arrays NA and NB to find which nodes are connected. If an adjacent node is found, the routine returns to subroutine SOLVE with IROW pointing to the location of the conductor in array G. Again, the documentation in subroutine SEARCH is sufficient to explain it's function and use.

The floating point processor routines were also written in assembler. This software interfaces the Fortran programs to the Intel 8232 FPP giving significant increases in execution speed. The multiply, addition, subtraction, and division routines in the Fortran library are named $MB, $AB, $SB and $DB respectively. These routines have the same names in the FPP library and they replace the Fortran versions when all the subroutines are linked together giving approximately 10 times faster arithmetic. The subroutines ELOG, ESQRT and CBRT perform the functions of natural algorithm, square root and cube root. These routines run approximately 23 times faster than the Fortran versions. These 7 subroutines represent the most heavily used arithmetic functions in the thermal analyzer program. The interested reader is referred to Reference (28) for more information on the algorithms used in these routines.

Numerous other subroutines are required to run the thermal analyzer program. As these subroutines were developed and debugged they were compiled and placed into either the thermal library or the Fortran utilities library of relocatable object modules. Most of these routines deal with adjusting the parameters in the thermal math model. Others, such as GTTIME which is used to obtain the current time of day from the system clock, are general utility subprograms. If one were to run the thermal analyzer on another computer system, the documentation in each routine should be read to determine if the routine is applicable.

A number of variables in the common block CONST are used to control the operation of the program. The variables and their functions are:

NNOD - the total number of nodes in the thermal network

NCOND - the total number of conductors in the thermal network

NAMSIZ - the number of four byte locations used to store the name of the run in ANAME

NLOOP - the maximum number of iterations allowed on the thermal network

LOOPCT - the actual number of iterations performed

OUTPUT - the output interval. This variable defines when subroutine PRNOUT is called

TIMEM - the mean time = (current time - last time)/2

TIMEND - the time at which the program is to terminate

TIMEO - the previous or old time

DTIMEI - the user specified time step used to advance the program through the simulation

DTIMEU - the time step actually used

DRLXCA - the allowable maximum temperature change between iterations. Used for determining if the iterations have converged to sufficient accuracy

DRLXCC - calculated maximum temperature change. DRLXCC for all nodes must be less than DRLXCA.

NDRLXC - the node with the largest temperature difference between iterations. Useful for debugging the thermal network

TIMEN - the current time in the simulation run. Interpolation routines use this value for calculating the new boundary conditions.

All of these values are set in the BLOCK DATA subroutine. Note that time, as used above, means the time in the simulated run.

5.0 THERMAL MATH MODEL

This section presents the solar energy system selected for this work, discusses how it is thermally modelled, and gives the model's nodal breakdown and conductor layout. The input data and boundary conditions are also presented. The controller software is discussed at the end of this section.

5.1 General

After the thermal analyzer program has been developed, the solar collector system must be selected, modelled and its parameters input to the program. Recall that the purpose of this work is to compare solar energy collector system performance using two different controllers. Therefore the system modelled need only approximate a typical real solar energy collection system. As long as both controllers interact with the same model then the comparison between the two controllers will be valid.

The solar collector system selected for this work consists of a set of parallel flow flat plate water cooled collectors connected to a water storage tank. A pump, controller, and piping delivers water from the storage tank to the collectors and then back to the storage tank. This closed system uses no heat exchangers and, for simplicity, assumes that water is always in the collectors, i.e., the water contains a substance to prevent freezing so that a drain down is not required. Heat losses from the piping and

storage tank are considered negligible. The collectors are south facing and inclined at an angle of 70° to the horizontal. The collectors are double glazed flat plate water cooled collectors bounded at the back by an attic (i.e., the collectors are mounted on the roof of a house). The controller compares collector outlet water temperature and storage tank temperature to determine if the water flow rate must be adjusted. This system is typical of the kind that would be installed for space heating in Winnipeg, Manitoba (Latitude 50°). Figure 2 shows the solar energy collection system used in this work.

The collector chosen for this work is the PPG Standard Aluminum Collector (the manufacturers data is included in Appendix IV). The author gained experience with this collector while working on the solar demonstration project on the Manitoba Legislative Buildings in Winnipeg(19). The author believes this collector is well designed and constructed and is also representative of water cooled collectors.

5.2 Node Breakdown

The system shown in Figure 2 must be broken down into nodes in order to predict temperatures using the thermal analyzer program. To reduce program execution time the number of nodes in the model must be kept to a minimum. Therefore a scaled down system has been modelled which consists of one collector, a controller, a pump and a storage tank. Since the collector is symmetrical about its longitudinal centreline, a further reduction

in the number of nodes may be realized by modelling only one half of the collector. Also, large nodes are used where the temperature gradients are small thus keeping the number of nodes to an absolute minimum without sacrificing accuracy, calculation of capacitances and conductances are by the methods outlined in Section 3.3. The reader is referred to Appendix V which gives a complete listing of the thermal math model data.

The collector modelled for this work is the PPG Standard Aluminum Collector (see the manufacturers data in Appendix IV). Basically, this collector consists of an aluminum absorber plate, painted black, with integral tubing bounded on the top by two glass cover plates. The back of the absorber is covered with fibreglass insulation.

The absorber plate was modelled by a large number of nodes (total of 56) to maintain accuracy (see Figure 3). Each absorber node had solar energy impressed upon it in subroutine PRE by interpolating the solar boundary data, taken from the Solar Demonstration Project(20), and then calling subroutine STFLUX which multiplies the flux times the node's area and the solar absorbtivity of the paint and stores the result in the Q array.

The absorber nodes are coupled to each other by conductances. Figure 4 illustrates typical absorber nodes that are conductively joined. Edge losses are very small because: a) the absorber plate is separated from the edge channel (i.e.,

the collector frame) by insulating tape and b) there is little contact pressure between the absorber plate and the edge channel. All of the absorber plate nodes are conductively coupled through the fibreglas backing to the attic node. The attic node temperature was set (arbitrarily) to be 20°F above the ambient temperature. Note that, because of the high thermal resistance from the absorber plate to the attic node, the attic temperature does not have to be known with great accuracy. All of these conductors are constants, that is, they do not change with respect to time. Conductor values were pre-calculated (using the methods of Section 3.3) and put into file CLCTRDAT.DAT in the conductor data block.

The absorber nodes are also connected to the inside glass cover plate by radiation (calculated by STRADK) and convection conductances (calculated by STPLHC). Radiation form factors between the absorber plate and the inside glass cover plate were assumed to be 1.0 because the two plates are very closely spaced. For the same reason, radiation and convection was assumed to occur only between nodes which are directly opposite each other.

Nodes describing the glass cover plates are much larger than the absorber nodes in order to reduce computation time. The glass cover plates are largely isothermal and therefore little accuracy is lost by using large nodes. The inner glass cover consists of 12 nodes while the outer glass plate consists of only 3 nodes. Convection conductances between these plates is set by STPLHC

in subroutine PRE. Convection conductance from the top cover plate to the ambient is calculated by STPLHC using the wind speed and direction. Radiation also occurs between the top cover plate and the sky. The value for this conductance is set by subroutine STRADK.

Most of the heat collected by the absorber nodes is transferred to the water in the tubes. Forty-eight nodes, representing the water in the tubes, are coupled to the absorber plate nodes via subroutine STTUBK. Subroutine STFLOW sets the fluid node to fluid node conductance.

The storage medium consists of 450 lbm of water in a storage tank. For simplicity, it is assumed that the tank is perfectly insulated and that the only heat transfer that occurs is via the pipes connecting it to the solar collector. Subroutine STFLOW couples the tank node to the inlet and outlet nodes of the collector.

All of the nodes which have been discussed so far have been diffusion nodes, i.e., they can store heat because they have capacitances. A second type of node, boundary nodes, have zero capacitance. These nodes consist of the ambient, the equivalent sky temperature, and the attic temperature. Temperatures for these nodes are interpoated in subroutine PRE from data taken from the Winnipeg International Airport, Meteorological Station.

5.3 Controllers

Two types of controllers are compared in this work. The first type is an on/off controller which is capable of circulating water to the collector at only two flow rates: zero or the pump's maximum capacity. The second type of controller can circulate water at an infinite number of flow rates varying from zero to the pump's maximum capacity.

Both controllers sense the difference between the collector outlet temperature and the storage tank to determine the flow rate. For the on/off controller, a temperature hysteresis was included in the algorithm to prevent rapid cycling of the pump. The variable flow rate controller algorithm varied the flow rate linearly over a small temperature difference range. Subroutine PRE contains both of these algorithms.

A wide range of flow rates were used to establish the performance of each controller. Five maximum flow rates of 500, 250, 125, 62.5 and 31.25 lbm/hour (135, 67.5, 33.7, 16.9, 8.4 Kg/hr-m^2) were set for each controller. There are seven days to analyze (see Section 5.4), five flow rates, and two controllers, therefore a total of seventy simulations have been run.

5.4 Boundary Conditions

The boundary conditions for this work were taken from the Solar Demonstration Project on the Legislative Buildings in Winnipeg, Manitoba(20) and from the Winnipeg International Airport,

Meteorological Station(20),(22). The solar flux on a south facing plane inclined at an angle of 70° to the horizontal was taken from the data collected on the Solar Demonstration Project. The ambient temperature, wind speed, and wind direction were taken from the airport data.

These data were interpolated in subroutine PRE, using time as the independent value, and then input to the thermal model. Heat flux values were interpolated and applied to the absorber plate nodes. Temperature and wind data were interpolated and applied to ambient, sky and attic temperatures and to calculate certain conductances.

The boundary data required to run the thermal analyzer program for the following seven days are included in Appendix V.

TABLE 1

RUN CODES

Run Code	Date
77231	August 19, 1977
77377	December 3, 1977
77347	December 13, 1977
77360	December 26, 1977
78001	January 1, 1978
78002	January 2, 1978
78007	January 7, 1978

Data for these dates provides a wide variation in boundary conditions which allow a good comparison to be made between the two types of controllers.

6.0 RESULTS

Because of the large amount of data produced from the seventy simulations (there are over 1.4 million temperatures) the data have been reduced to graphical form. The types of graphs plotted are:

(a) Boundary Data

i) Solar Insolation vs. Time
ii) Ambient Temperature vs. Time
iii) Wind Speed vs. Time
iv) Wind Direction vs. Time

Note that in iii) above North is 0 radians, West is 1.57 radians.

(b) Simulation Data Output

i) Flow Rate vs. Time
ii) Instantaneous Energy vs. Time
iii) Efficiency vs. $(T_{av}-T_{amb})/I$
iv) Temperature vs. Time

A typical set of plots for December 3, 1977 is given in Figures 5 to 48.

Ten points were plotted for each hour of simulated operation giving 100 data points per curve in the Figures 5 to 48. The curves were made smooth by using subroutine DCSINT(27) to fit cubic polynomials to the data.

The Flow Rate vs. Time graphs show how the controller is reacting. For the on/off controller change from zero flow rate to the maximum flow rate indicates that the pump has been commanded on by the controller. Conversely, a change from the maximum flow rate to zero indicates the pump has been turned off. Cycling of the pump is shown (especially in Figure 25) by a series of spikes. The same holds true for the variable flow rate controller except that the transition is much more gradual.

The on/off controller subtracts the tank temperatures from the collector plate temperature near the exit manifold and compares this result with the on and off set points. When this difference exceeds 12.6°F (7.0°C) the controller turns on the pump which then forces water through the collector. If the tank and collector temperature difference is less than 5.4°F (3°C) the pump is turned off and the flow rate through the collector is zero. The spread between the on and off set points is the hysteresis built into the controller which reduces cycling of the pump. The Flow Rate vs. Time graphs shown in Figures 9, 13, 17, 21 and 25 illustrates an increasing tendency to cycling as the maximum flow rate is increased. This is because the collector plate temperature is held closer to the tank temperature as the flow rate is increased. Therefore the temperature difference between the collector plate and the tank is much closer to the on and off set points thereby causing cycling of the pump when there are changes in the solar flux. This causes

a decrease in collected energy because the flow rate is zero while the controller waits for the collector temperature to rise. Evidence of this can be seen in Figures 51 to 57 by the drop in daily efficiency at the higher flow rates for the on/off controller.

The variable flow rate controller adjusts the flow rate linearly between the same limits (see Subroutine PRE in Appendix I). The result is that the flow rate can be seen (by overlaying the graphs) tracking the solar flux. Thus, cycling is avoided and the temperature of the absorber plate is maintained closer to the tank temperature at the higher flow rates. Note that at low flow rates the profile of the Flow Rate vs. Time curves (Figures 9, 13, 29 and 33) are almost identical since both controllers are operating at the maximum flow rate. This accounts for their similar performance at low flow rates in Figures 51 to 57.

The Instantaneous Energy vs. Time graphs compare the amount of energy collected against the amount of energy available over the six minute time period (i.e., simulated time) between printouts. Figures 10, 14, 18, 22 and 26 generally show a larger amount of the available energy is collected as the maximum flow rate is increased. A minor exception to this occurs at the higher flow rates when the on/off controller is cycling the pump rapidly. Rapid cycling of the pump produces incorrect readings for the outlet to completely flow through the collector. This does not affect the calculation of daily efficiency (the primary objective of this work) because the storage tank

still receives the correct mass and temperature of water. To compensate for this problem on the graphs the collector instantaneous energy curves were drawn by hand, omitting those parts where the pump was cycling rapidly.

The Efficiency vs. $(T_{av}-T_{amb})/I$ curves illustrate the collector efficiency as a function of average absorber plate temperature ambient temperature and solar insolation. Note that the efficiency tends to increase as flow rate increases. Pump cycling causes scattering of the data for the on/off controller at higher flow rates (especially in Figure 27) because of difficulty in determining the average plate temperature T_{av} (which is found by area averaging all the absorber plate nodes in the thermal model). Again, this is caused by rapid pump cycling which does not allow the water to completely flow through the collector before a "snapshot" of the temperature profile is taken.

The Temperature vs. Time graphs are included to indicate the effect the controller and the flow rate have upon collector temperatures. Recall that by maintaining a small absorber plate to ambient temperature difference, the heat losses from the collector are decreased. For both the on/off and the variable flow rate controller, the collector plate temperatures are reduced as the flow rate increases. Note that at the higher flow rates (Figures 20 and 48) the absorber plate temperature is maintained very close to the tank temperature. The on/off controller however must cycle rapidly to accomplish this. The small peaks and valleys in the temperature profile, due to the

high flow rate and the changes in solar flux, of the absorber plate represent periods when energy is not collected because the controller has turned the pump off and is waiting for the temperature to rise again. The variable flow rate controller simply adjusts the flow rate to maintain a constant temperature difference between the collectors and the tank, therefore, the collector temperature profile of Figure 48 is much smoother. This corresponds to a greater amount of energy collected than by the on/off controller.

Figures 49 and 50 are 3-dimensional plots showing the spatial vaiation of temperatures on the absorber plate. Note that the ambient temperature is not to scale. Figure 49 is a "snapshot" of the absorber plate temperatures taken just after water had started flowing through the collector. Water entered at the bottom (lower left hand side of the graph) centre and began moving through the collector. Figure 50 shows the absorber plate temperature at 13:00 hours. At this time the water flow is well established. The depression at the bottom centre of the collector clearly shows where the water enters the collector. In both diagrams the non-isothermal nature of the absorber plate can be easily seen.

The author believes that by treating the solar collector as a non-isothermal plate and employing the basic heat transfer equations in a finite difference model of a solar collector, an accurate comparison between the two types of controllers

has been achieved. The techniques given in Reference 7 usually treat the solar collector as an isothermal plate and use many simplifying assumptions when analyzing a solar energy system. Although the techniques of Reference 7 are quite suitable for overall design purposes, the author believes that much more detailed methods are needed when analyzing the effects of controller systems which require continuous, accurate sensor data in order to operate correctly. Computer simulation methods that use a finite difference approach can achieve the goal of greater accuracy especially in transient cases.

The collector efficiencies predicted by the computer program can be compared with the manufacturer's test data given in Appendix IV. The simulation data used in this comparison was from day 77.337 with a flow rate of 62.5 lbm/hr (16.9 Kg/hr m^2) and an ambient temperature of -10°F (-23°C). The manufacturer's field test data is plotted in Figures 10, 11 and 12 of Appendix IV. Figure 10 was used for this comparison, because the ambient temperature more closely approximates the ambient temperature used in the computer program than either Figures 11 or 12. By comparing Figures 10 and 11 of Appendix IV it can be seen that as the ambient temperatures decreases from 100°F (38°C) to 50°F (10°C), the efficiencies are raised by about 5 to 15% when T = 100°F (38°C) and the solar insolation is between 150 and 200 Btu/hr-ft^2 (473 to 630 W/M^2°C). Therefore it would be expected that the computer program (which for day 77.337 used an ambient temperature of -10°F (-23°C)) would predict efficiencies 5 to 15% less than those predicted by Figure 10 in Appendix

IV. To compensate for this, the efficiencies in Figure 10 were corrected by adding the difference in efficiencies between Figures 10 and 11 at the same ΔT and solar insolation. Since the difference in ambient temperatures between Figures 10 and 11 was 50°F (-23°C), then the corrected efficiencies would approximate efficiencies at an ambient temperature 0°F (-18°C).

Table 2 presents data taken from the simulation of day 77.337. The controller was a variable flow rate type and the maximum flow rate was set at 3.47 lbm/hr-ft^2 (16.9 Kg/hr-m^2). Note that from 10:00 to 15:00 hours the flow rate was constant at its maximum rate. Collector efficiencies were calculated from the simulation data using the following formula:

$$n = \frac{\dot{m}\, C_p\, (T_{fo} - T_{fi})}{I} \qquad (26)$$

where

$$\dot{m} = 3.47 \text{ lbm/hr-ft}^2 \text{ (16.9 Kg/hr-m}^2\text{)}$$

and

$$C_p = 1.0 \text{ Btu/lbm°F (4188 J/s-Kg-°C)}$$

Some differences between the predicted and the corrected measured efficiencies were expected because the computer program was simulating a transient collector system with widely fluctuating solar insolation (see Figure 5) instead of a steady-state system with a fairly constant solar

TABLE 2

EFFICIENCY COMPARISON

TIME	SOLAR INSOLATION Btu/Hr-Ft^2	(w/m^2)	AMBIENT TEMPERATURE °F	(°C)	FLUID OUTLET TEMPERATURE °F	(°C)	FLUID INLET TEMPERATURE °F	(°C)	EFFICIENCY FROM FIG.10 APPENDIX IV T_{amb}= 50°F(10°C)	CORRECTED EFFICIENCIES FROM FIG.10 APPENDIX IV $T_{amb} \approx$ -10°F(-23°C)	EFFICIENCY PREDICTED BY PROGRAM T_{amb}= -10°F(-23°C)
10:00	183	(579)	-9.9	(-23.3)	101.5	(38.6)	80.9	(27.2)	35	42	38.6
11:00	190	(602)	-8.1	(-22.3)	114.6	(45.9)	84.4	(29.1)	38	44	54.7
12:00	224	(708)	-6.9	(-21.6)	120.0	(48.9)	89.0	(31.7)	40	46	47.6
13:00	184	(582)	-5.4	(-20.8)	124.3	(51.3)	94.1	(34.5)	32	38	56.6
14:00	196	(619)	-4.4	(-20.2)	127.2	(52.9)	98.4	(36.9)	33	40	50.6
15:00	125	(394)	-4.4	(-20.2)	113.7	(45.4)	101.5	(38.6)	18	33	34.0

flux. However, the corrected measured efficiencies and the predicted efficiencies do correlate reasonably well.

Figures 9 to 48 can be reduced further by plotting the daily efficiency as a function of flow rate and the controller type. The daily efficiency is calculated as the increase in the amount of energy stored in the tank at the end of the day divided by the total amount of solar energy available for that day. The flow rates correspond to the maximum flow rate set for each type of controller. The seven graphs, one for each day analyzed, relating daily efficiency and flow rates, are given in Figures 51 to 57.

These graphs consistently and clearly show that the variable flow rate controller does not offer a significant increase in performance at flow rates less than 13 lbm/hr-ft^2 (65 kg/hr-m^2). Only at higher flow rates, where the energy losses due to cycling of the pump for the on/off controller becomes apparent, does the variable flow rate controller have the advantage.

Determination of the optimum flow rate is beyond the scope of this work, however, some general comments may be made based on Figure 49 to 55. The daily efficiency curve flattens out rapidly after about 4 lbm/hr-ft^2 (20 kg/hr-m^2). Further increases in flow rate gain only a few percent in efficiency while dramatically increasing pumping power. Therefore, the optimum flow rate would appear to lie between 4 and 8 lbm/hr-ft^2 (20 and 40 kg/hr-m^2). To determine the optimum flow rate one would have to perform an energy balance on the solar energy system, with special consideration given to the pumping energy output versus the energy collected.

7.0 CONCLUSIONS

This work has shown that the on/off controller and the variable flow rate controller provide virtually identical solar energy system daily efficiencies at flow rates less than 8 lbm/hr-ft^2 (40 kg/hr-sqm). Beyond this the variable flow rate controller offers only a slight improvement in efficiency. But, at the higher flow rates, erosion corrosion and pumping power become important considerations decreasing the advantage offered by the variable flow rate controller. Based on these results the optimum controller configuration appears to be the generally less expensive on/off controller set for a maximum capacity between 4 and 8 lbm/hr-ft^2 (20 and 40 kg/hr-sqm). The solar energy system designer should be aware of the limited advantages of the variable flow rate controller in order to properly assess its suitability.

The Z-80A microcomputer system used in this work was essential.With a fast Floating Point Processor (FPP) installed in the microcomputer and by using the author's specially designed FPP routines which utilize parallel processing techniques, this microcomputer system was capable of performing the 70 simulations in two weeks of 24 hours a day running. Without the FPP and the special software, the execution of the 70 simulations would have easily exceeded 20 weeks of 24 hours a day running. If the microcomputer system itself was not available then the computer costs for the simulations would have been more than $500,000; far beyond the resources allotted for this work. The capital cost of $10,000 for the microcomputer system are therefore easily justified.

The solar energy system model and the thermal analyzer program can be easily adapted to include additional parameters. By simulating a number of different solar energy system configurations, it would be possible to optimize the system without having to build up an actual solar energy system. If the computing time is inexpensive, then a large number of solar energy systems could be examined at little cost. However, the computer program should be verified first. The verification would consist of an actual solar energy system that would be closely monitored. This system would be built only once and used only for the purposes of verifying the computer program. After the data had been collected, the computer program would be run using climatic data obtained in the experiments. If the program and the experiment agree closely then parametric runs could be made with confidence. Otherwise some adjustment of the computer program may be necessary such as more detailed modelling of the convection currents between the glass cover plates or absorber plate. Note that detailed modelling of a solar collector is needed only when high accuracy, expecially in the transient mode, is needed. Normally the techniques listed in Reference 7 are sufficient for solar energy system design purposes.

8.0 REFERENCES

1. Kovarik, M., and Lesse, P.F. "Optimal Control of Flow in Low Temperature Solar Heat Collectors", Solar Energy, Vol. 18, pp.431-435, 1976.

2. Siebers, D.L., and Viskanta, R. "Thermal Analysis of Some Flat-Plate Solar Collector Designs for Improving Performance", Journal of Energy, Vol. 3, No.1, pp.8-15, 1979.

3. Grossman, G., Shitzer, A., and Zvirin, Y. "Heat Transfer Analysis of a Flat-Plate Solar Energy Collector", Solar Energy, Vol. 19, No.5, pp.493-502, 1977.

4. Prabhakar, P.R., Francis, J.E., and Love, T.J. "Two-Dimensional Analysis of a Flat-Plate Solar Collector", AIAA Paper No.76-451, 1976.

5. Arafa, A., Fisch, N., and Hahne, E. "Parametric Investigation of Flat-Plate Solar Collectors", Proceedings of the International Solar Energy Society Congress, New Delhi, India, January 16-21, 1978, pp.917-923.

6. Thomas, W.C., and Vaughan, D.H. "Thermal Performance of Flat-Plate Solar Collectors", Paper No.78-4528, Presented at the Winter Meeting of the ASAE, Chicago, Illinois, December 18-20, 1978.

7. Duffie, J.A., and Beckman, W.A. Solar Energy Thermal Processes, New York, New York, John Wiley and Sons Inc. 1974.

8. Rohsenow, W.M., and Hartnett, J.P. (Ed.), "Handbook of Heat Transfer", New York, New York, McGraw-Hill Inc, 1973.

9. Roecker, R.H. (Ed.), Heat Transfer Data Book, Schenectady, New York, Corporate Research and Development, General Electric Company, 1970.

10. Ferziger, J.H. Numerical Methods for Engineering Application, New York, New York, John Wiley & Sons, 1981.

11. Martin Marietta Corporation, Martin Interactive Thermal Analysis System, Denver, Colorado, 1971.

12. Krieth, F. Principles of Heat Transfer, 3rd Edition, New York, New York, Intext Press Inc, 1973.

13. Myers, G.E. Analytical Methods in Conduction Heat Transfer, New York, New York, McGraw-Hill Inc, 1971.

14. Cromemco Inc., Cromemco Fortran IV Reference Manual, Part No. 023-0038, Mountain View, California, Cromemco Inc. 1981.

15. Cromemco Inc., Cromemco Z-80 Macro Assembler Instruction Manual, Part No. 023-0039, Mountain View, California, Cromemco Inc. 1981.

16. Zilog Inc., Z-80, Z-80A Technical Manual, Cupertino, California, Zilog Inc, 1977.

17. Intel Corporation, "Intel Component Data Catalog", Santa Clara, California, Intel Corporation, 1980.

18. Shurcliff, W.A. Solar Heated Buildings of North America: 120 Outstanding Examples, Harrisville, New Hampshire, Brick House Publishing Co, 1978.

19. Chant, R.E., Bryant, D.R., and Gemmel, R. Operational Experiences with the Solar Demonstration Project on the Manitoba Legislative Building, Winnipeg, Manitoba. Paper presented at the 1977 General Meeting and Conference of the Solar Energy Society of Canada, Edmonton, Alberta.

20. Bryant, D.R., and Chant, R.E. Instrumentation and Data Processing for the Solar Demonstration Project, Legislative Building, Winnipeg, Mantioba. Paper presented at the 1977 General Meeting and Conference of the Solar Energy Society of Canada, Edmonton, Alberta.

21. Environment Canada, Atmospheric Environment "Monthly Radiation Summary", Atmospheric Environment Service, Government of Canada, Downsview, Ontario, August 1977, December 1977 and January 1978.

22. Environment Canada, Atmospheric Environment, "Monthly Meteorological Summary", Atmospheric Environment Service, Government of Canada, Downsview, Ontario, August 1977, December 1977 and January 1978.

23. Baumeister, T. (Ed.), "Marks' Standard Handbook for Mechanical Engineers", 7th Edition, New York, New York, McGraw-Hill Inc, 1967.

24. Selby, S. CRC Standard Mathematical Tables, Cleveland, Ohio, The Chemical Rubber Co. 1971.

26. Zaks, R., The CP/M Handbook with MP/M, Berkeley, California, Sybex Inc. 1980.

27. Duris, C.S. Fortran Routines for Discrete Cubic Spline Interpolation and Smoothing, Algorithm 547, New York, New York. The Association for Computing Machinery, Inc. 1981.

28. Cody, W.J., and Waite, W. Software Manual for the Elementary Functions, Englewood Cliffs, New Jersey, Prentice-Hall Inc. 1980.

29. Siegel, R and Howell, J.R. Thermal Radiation Heat Transfer, New York, New York, McGraw-Hill Inc, 1972.

30. Bayley, F.J., Owen, J.M. and Turner, A.B. Heat Transfer, Don Mills, Ontario, Thomas Nelson and Sons Ltd, 1972.

31. Gebhart, B. Heat Transfer, Toronto, Ontario, McGraw-Hill Book Company, 1961.

32. Eckert, E.R.G. and Drake, R.M. Analysis of Heat and Mass Transfer, Toronto, Ontario, McGraw-Hill Inc, 1972.

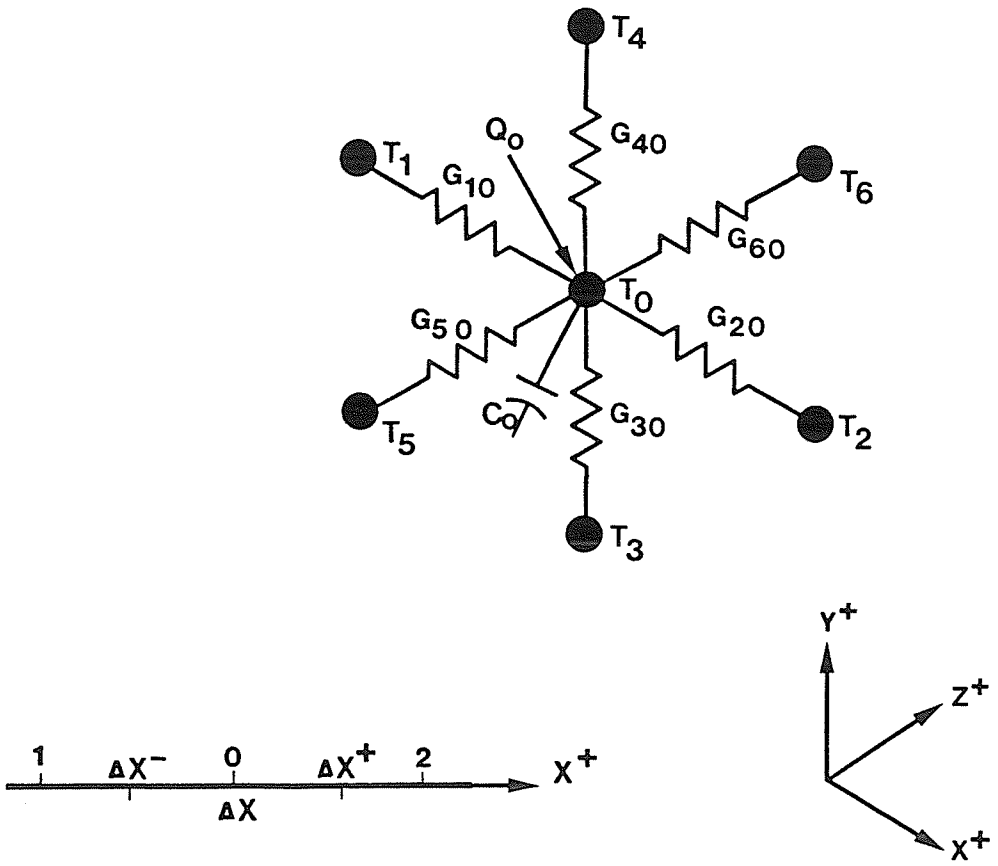


FIGURE 1: GENERAL NODE LAYOUT

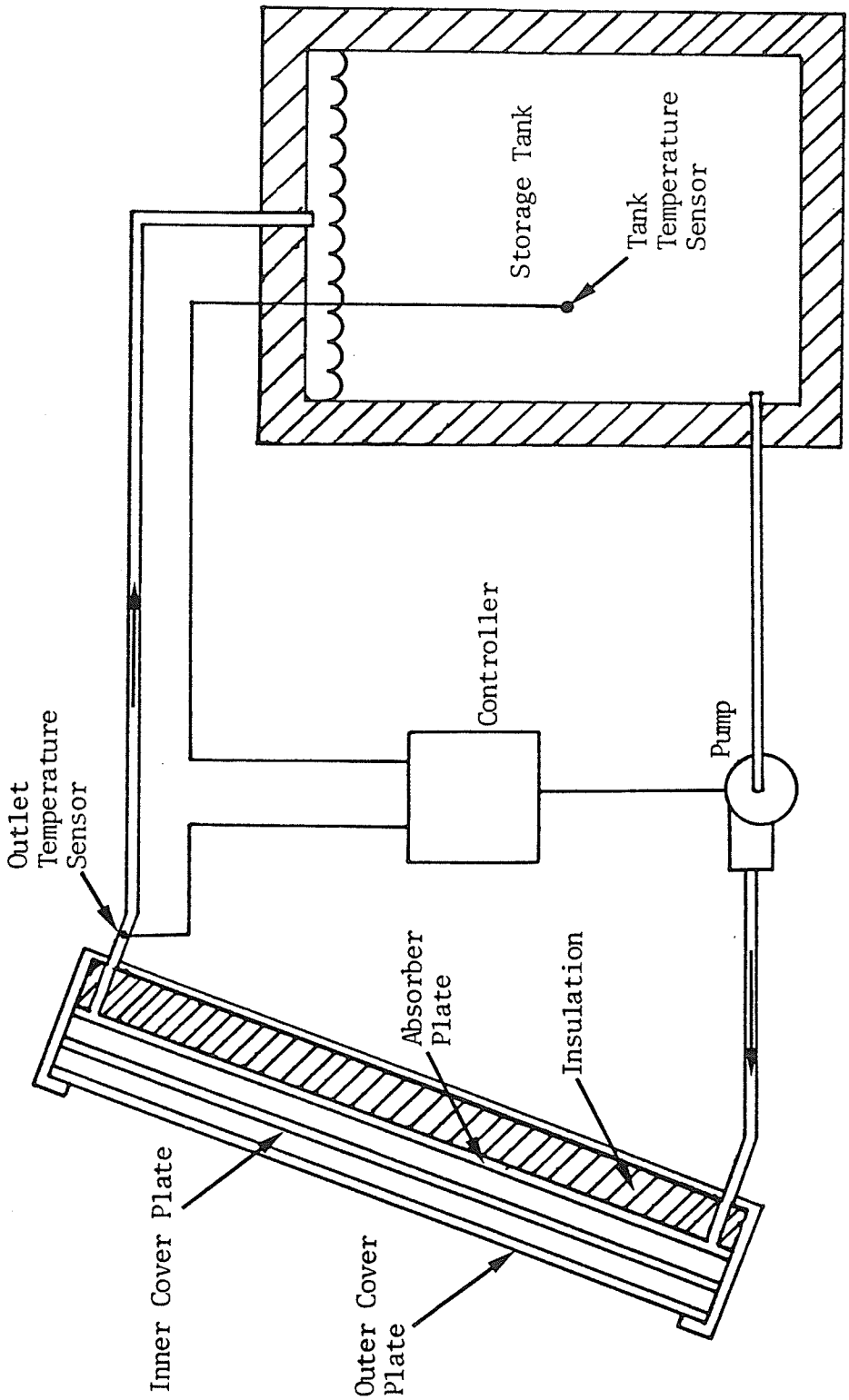


FIGURE 2: SCHEMATIC OF THE SOLAR ENERGY SYSTEM MODELLED

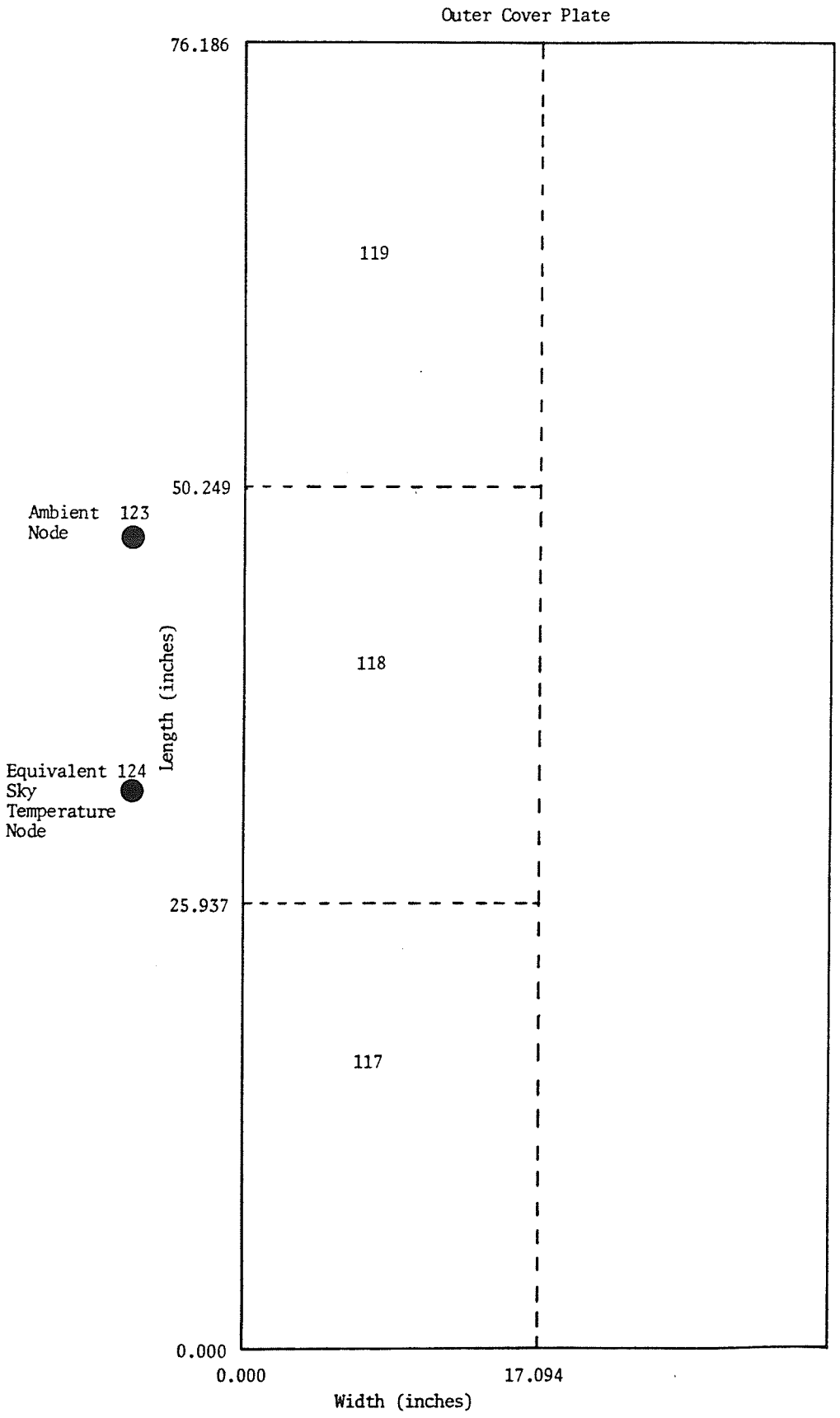


FIGURE 3: NODAL BREAKDOWN

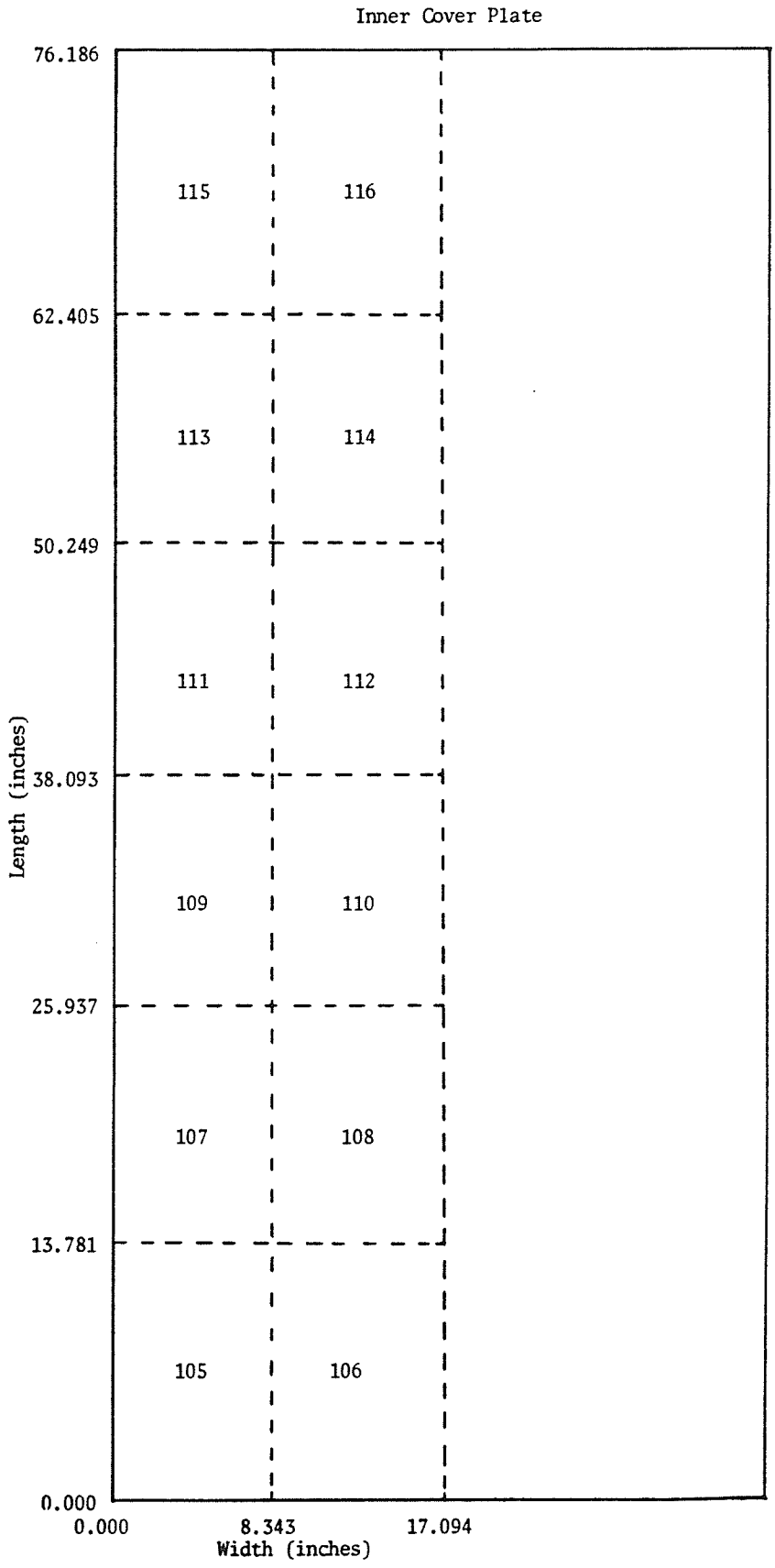


FIGURE 3: NODAL BREAKDOWN (continued)

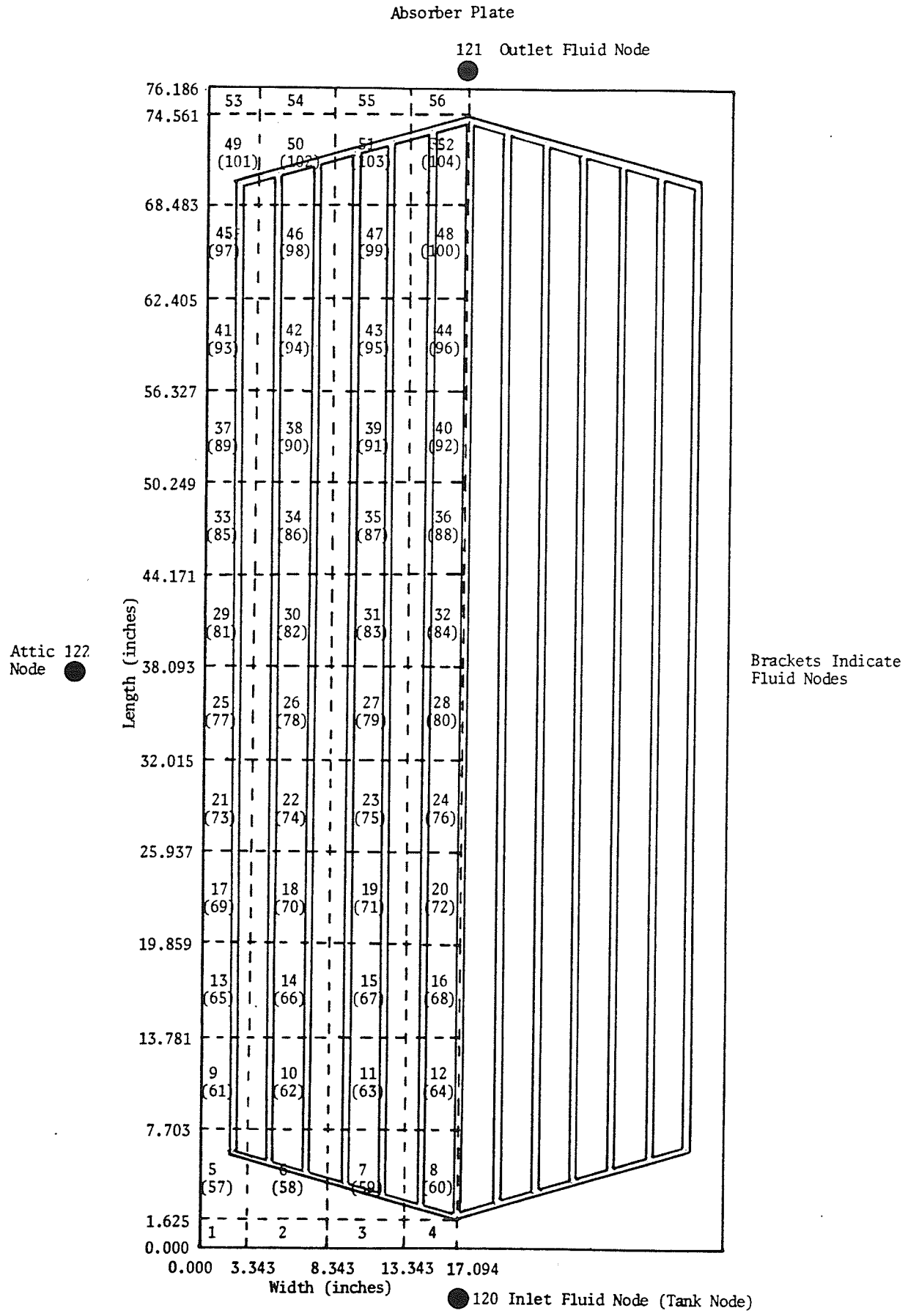


FIGURE 3: NODAL BREAKDOWN (continued)

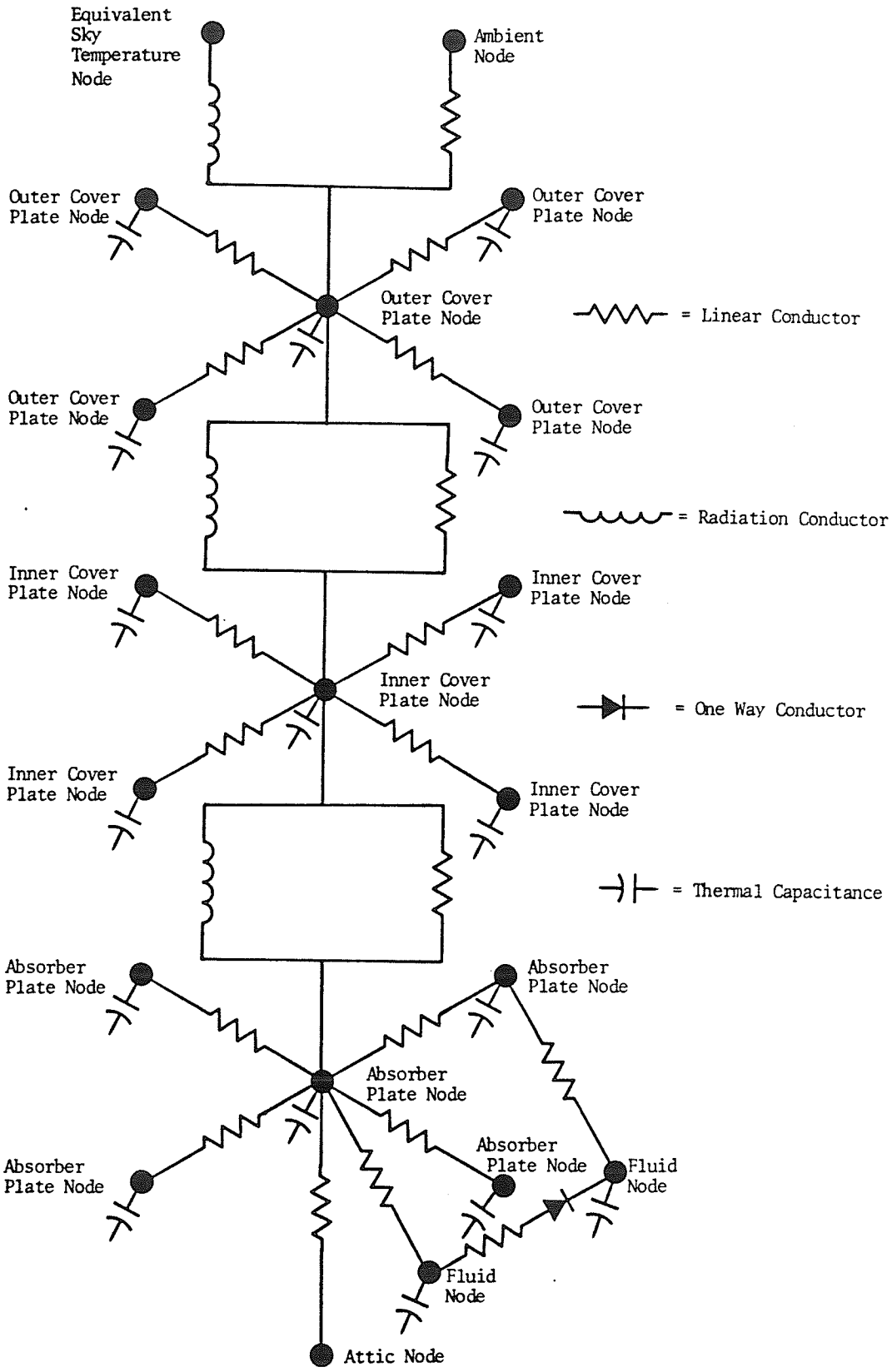


FIGURE 4: TYPICAL NODES AND CONDUCTORS

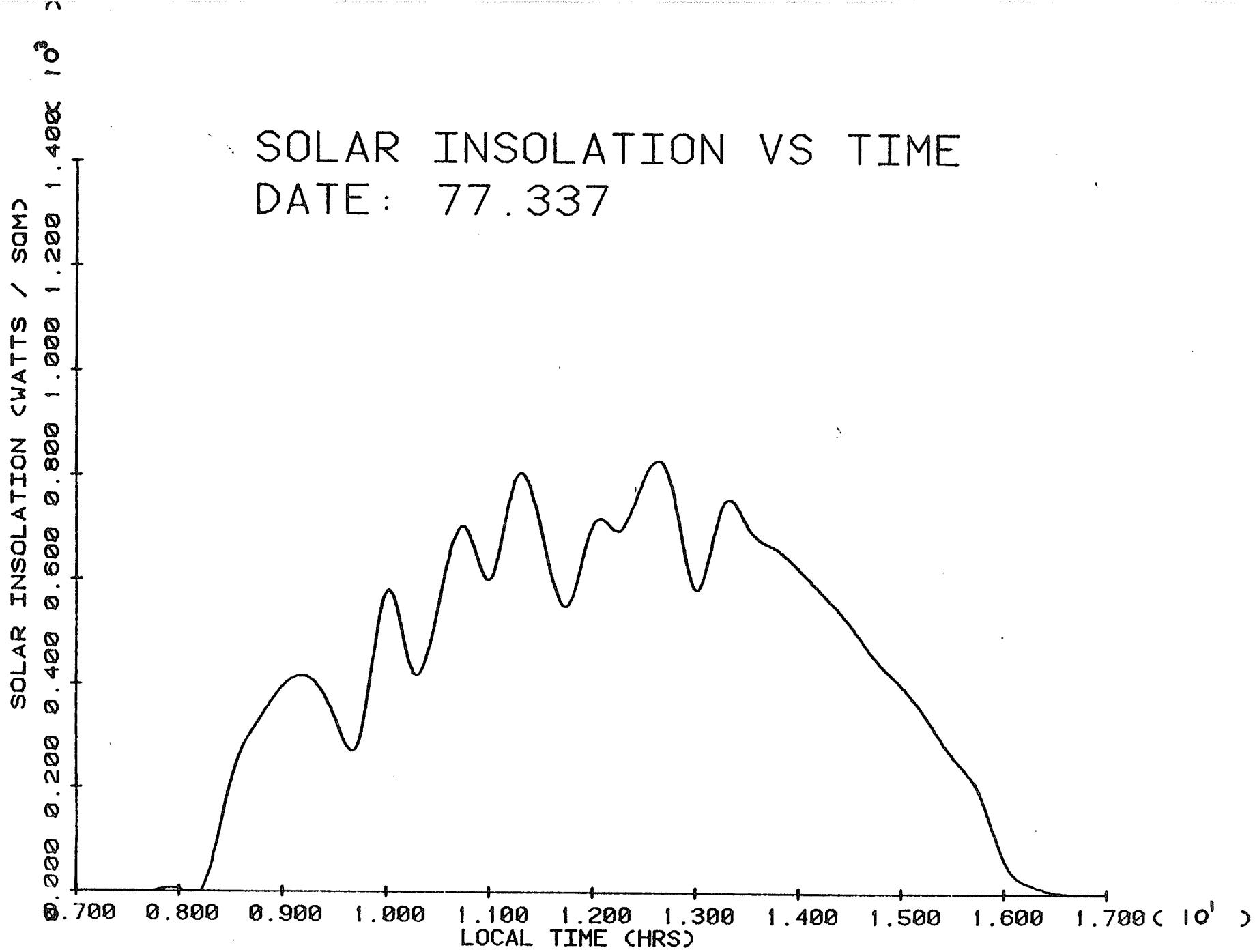

SOLAR INSOLATION VS TIME
DATE: 77.337
SOLAR INSOLATION (WATTS / SQM)
0.000 0.200 0.400 0.600 0.800 1.000 1.200 1.400 X 10^3
0.700 0.800 0.900 1.000 1.100 1.200 1.300 1.400 1.500 1.600 1.700 (10')
LOCAL TIME (HRS)

FIGURE 5

AMBIENT TEMPERATURE VS TIME
DATE: 77.337
AMBIENT TEMPERATURE (C)
-1.500.000
1.500
3.000
4.500
6.000
7.500
(10')
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
(10')
LOCAL TIME (HRS)

FIGURE 6

WIND SPEED VS TIME
DATE: 77.337
WIND SPEED (KM/HR)
0.000
0.500
1.000
1.500
2.000
2.500
3.000
3.500
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
LOCAL TIME (HRS)

FIGURE 7

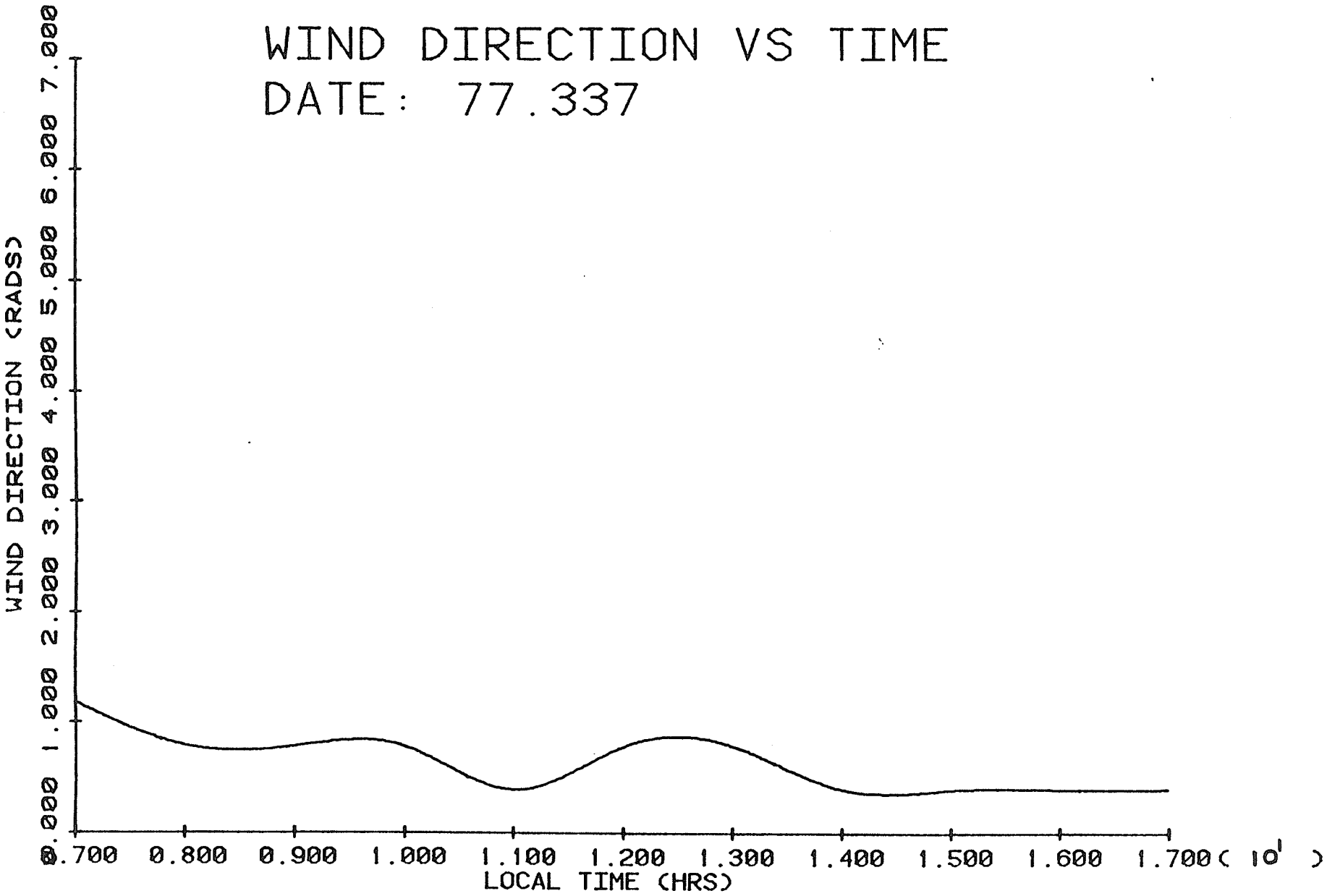

WIND DIRECTION VS TIME
DATE: 77.337
WIND DIRECTION (RADS)
0.000
1.000
2.000
3.000
4.000
5.000
6.000
7.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
(10)
LOCAL TIME (HRS)

FIGURE 8

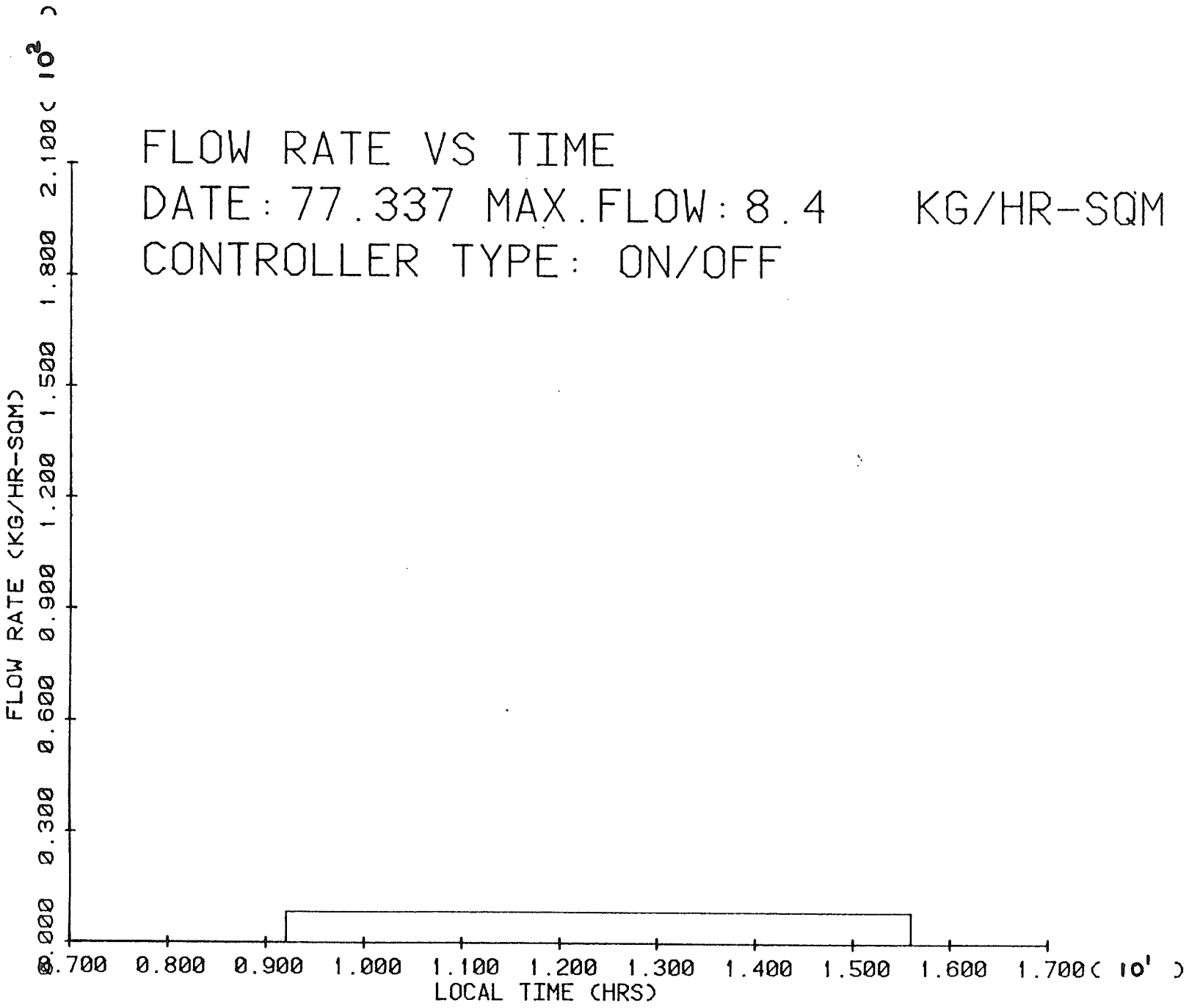

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
FLOW RATE (KG/HR-SQM)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100 (10^2)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10^1)
LOCAL TIME (HRS)

FIGURE 9

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX. FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
LOCAL TIME (HRS)

FIGURE 10

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
EFFICIENCY (PER CENT)
1.000
0.800
0.600
0.400
0.200
0.000
0.984 1.132 1.281 1.429 1.578 1.726 1.874 2.023 2.171 2.320 2.468 (10^-1)
(TAV-TAMB)/I
(10^2)

FIGURE 11

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
□ COLLECTOR PLATE
▲ INNER COVER PLATE
+ OUTER COVER PLATE
Y TANK
TEMPERATURE (C)
9.000
7.000
5.000
3.000
1.000
-1.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
LOCAL TIME (HRS)

FIGURE 12

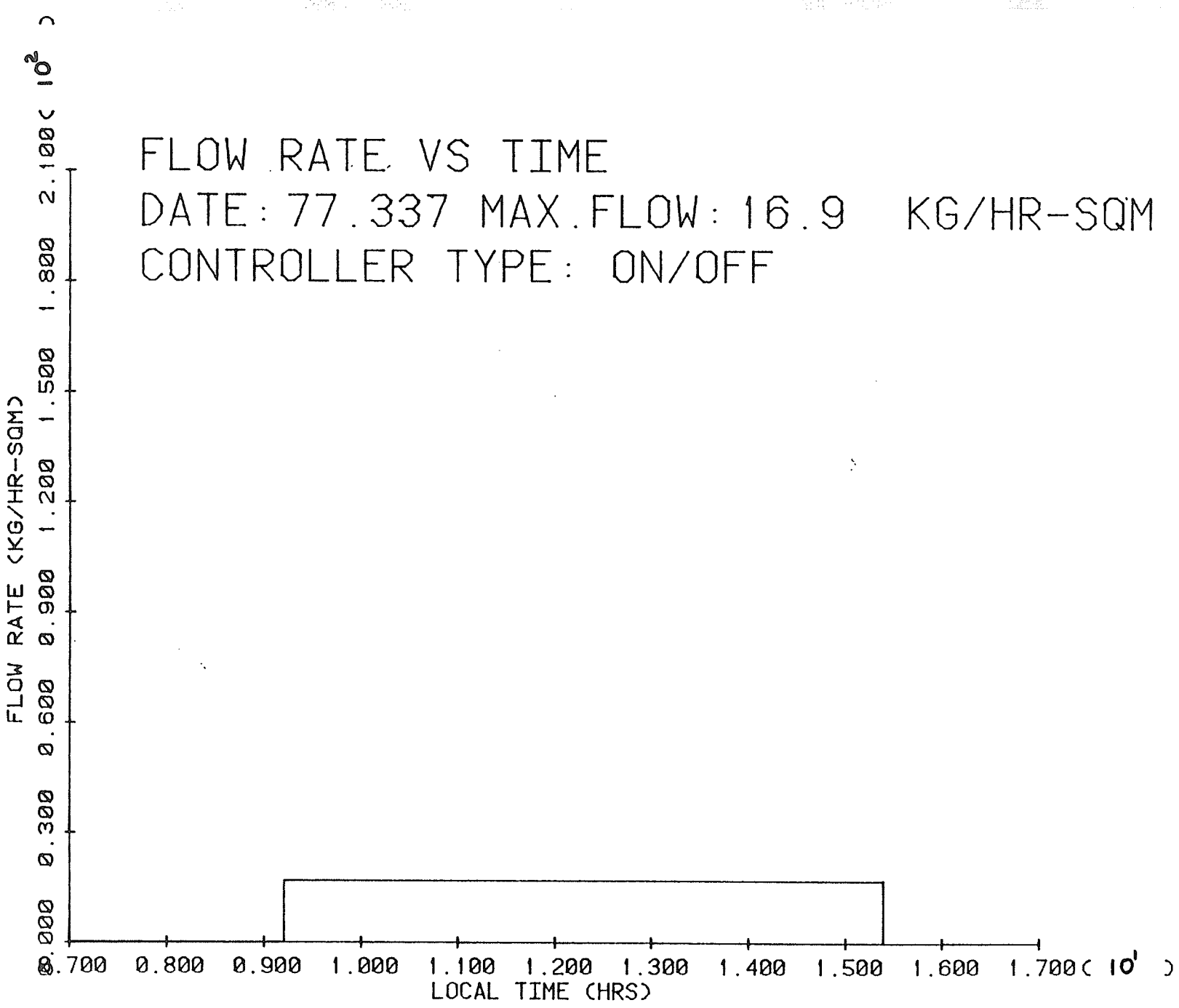

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
FLOW RATE (KG/HR-SQM)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100 (10^2)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10^1)
LOCAL TIME (HRS)

FIGURE 13

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
LOCAL TIME (HRS)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700

FIGURE 14

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
EFFICIENCY (PER CENT)
1.000
0.800
0.600
0.400
0.200
0.000
0.874
0.998
1.123
1.247
1.371
1.496
1.620
1.744
1.868
1.993
2.117
(TAV-TAMB)/I
(10²)
(10⁻¹)

FIGURE 15

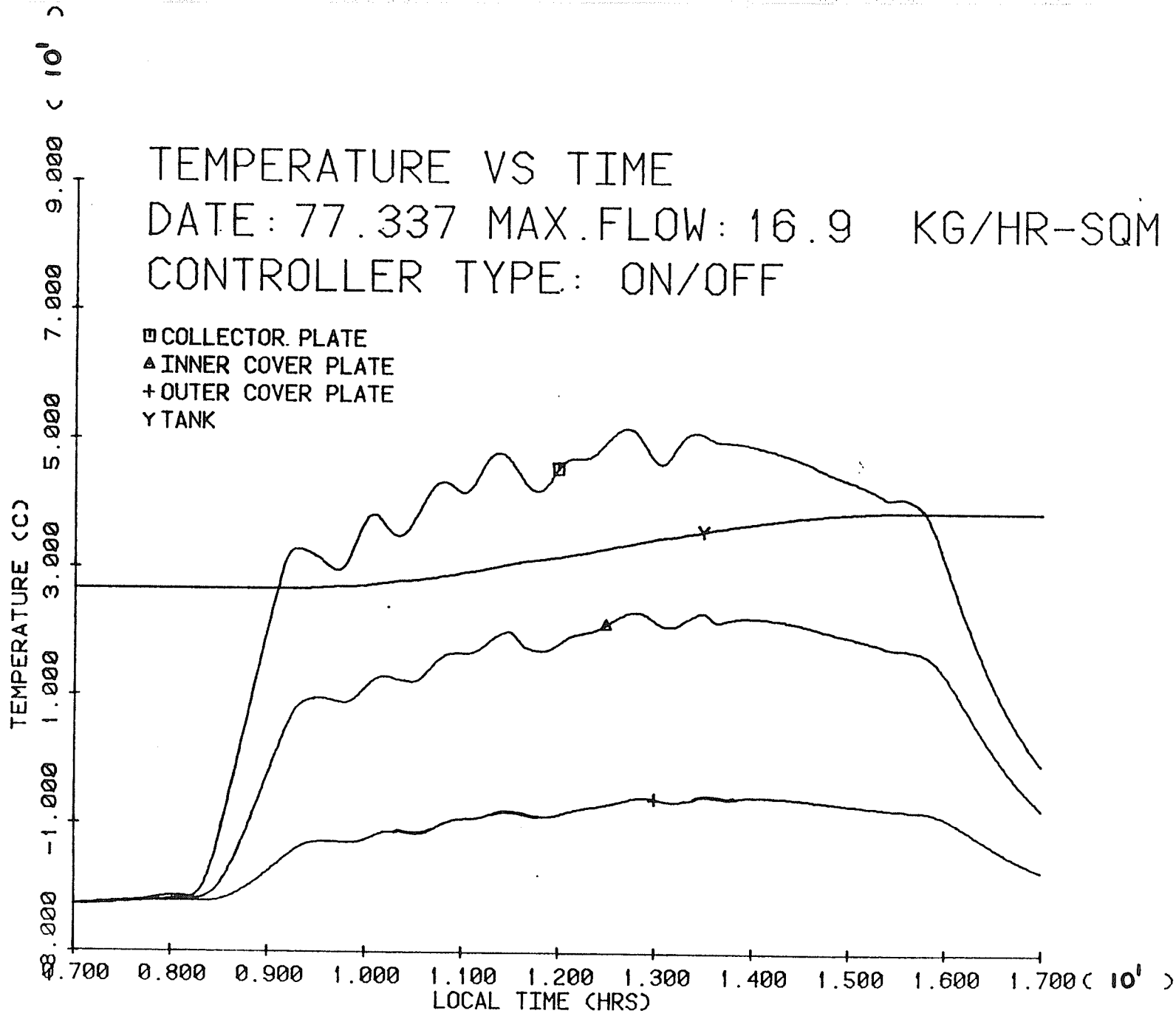

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
COLLECTOR PLATE
INNER COVER PLATE
OUTER COVER PLATE
TANK
TEMPERATURE (C)
9.000 (10')
7.000
5.000
3.000
1.000
-1.000
-3.000
0.700 0.800 0.900 1.000 1.100 1.200 1.300 1.400 1.500 1.600 1.700 (10')
LOCAL TIME (HRS)

FIGURE 16

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
FLOW RATE (KG/HR-SQM)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100 (10²)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 17

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
(10³)
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 18

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
EFFICIENCY (PER CENT)
1.000
0.800
0.600
0.400
0.200
0.000
0.793
0.913
1.033
1.153
1.273
1.393
1.514
1.634
1.754
1.874
1.994
(TAV-TAMB)/I
(10²)
(10¹)

FIGURE 19

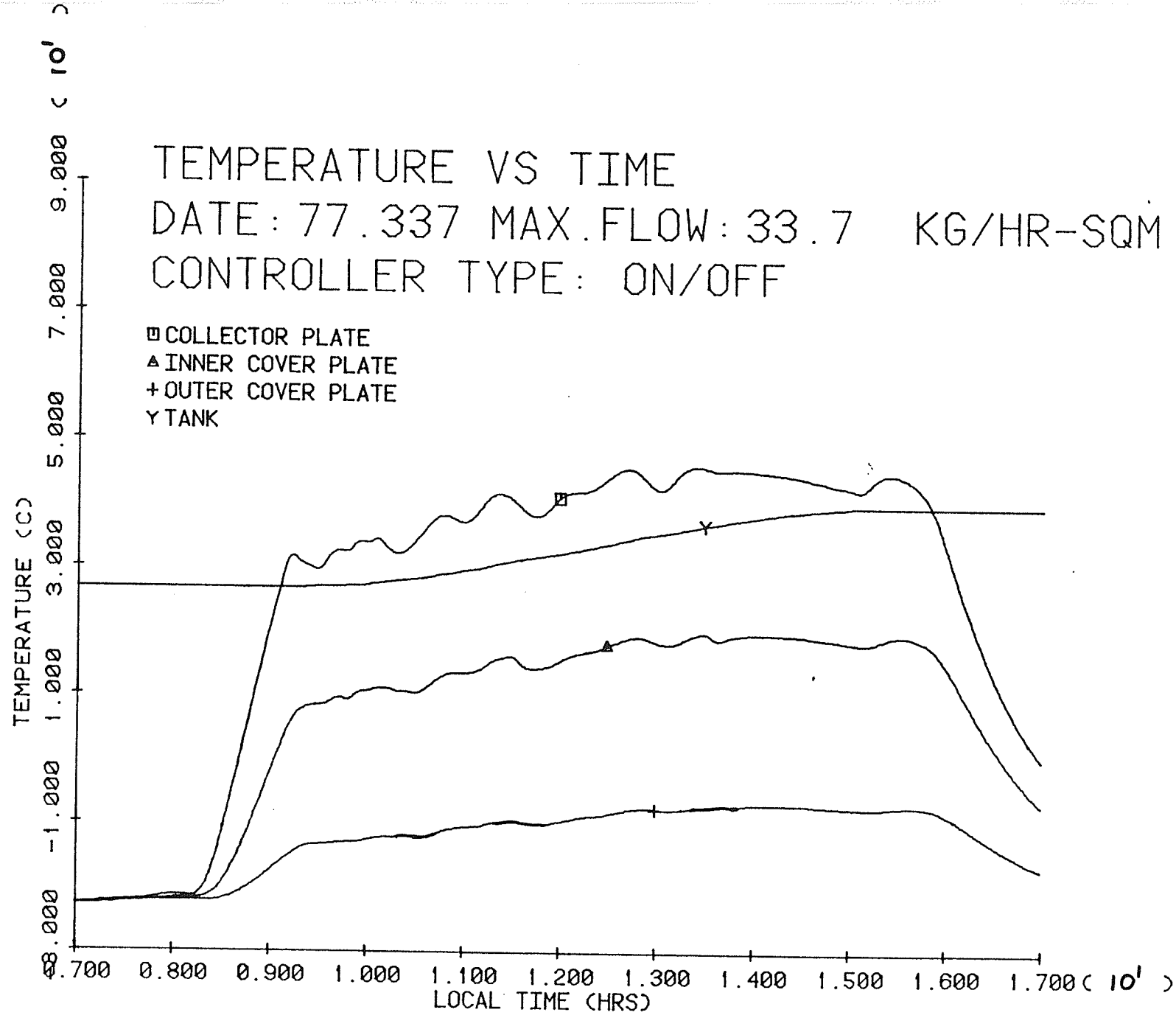

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
COLLECTOR PLATE
INNER COVER PLATE
OUTER COVER PLATE
TANK
TEMPERATURE (C)
9.000 (10')
7.000
5.000
3.000
1.000
-1.000
-3.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10')
LOCAL TIME (HRS)

FIGURE 20

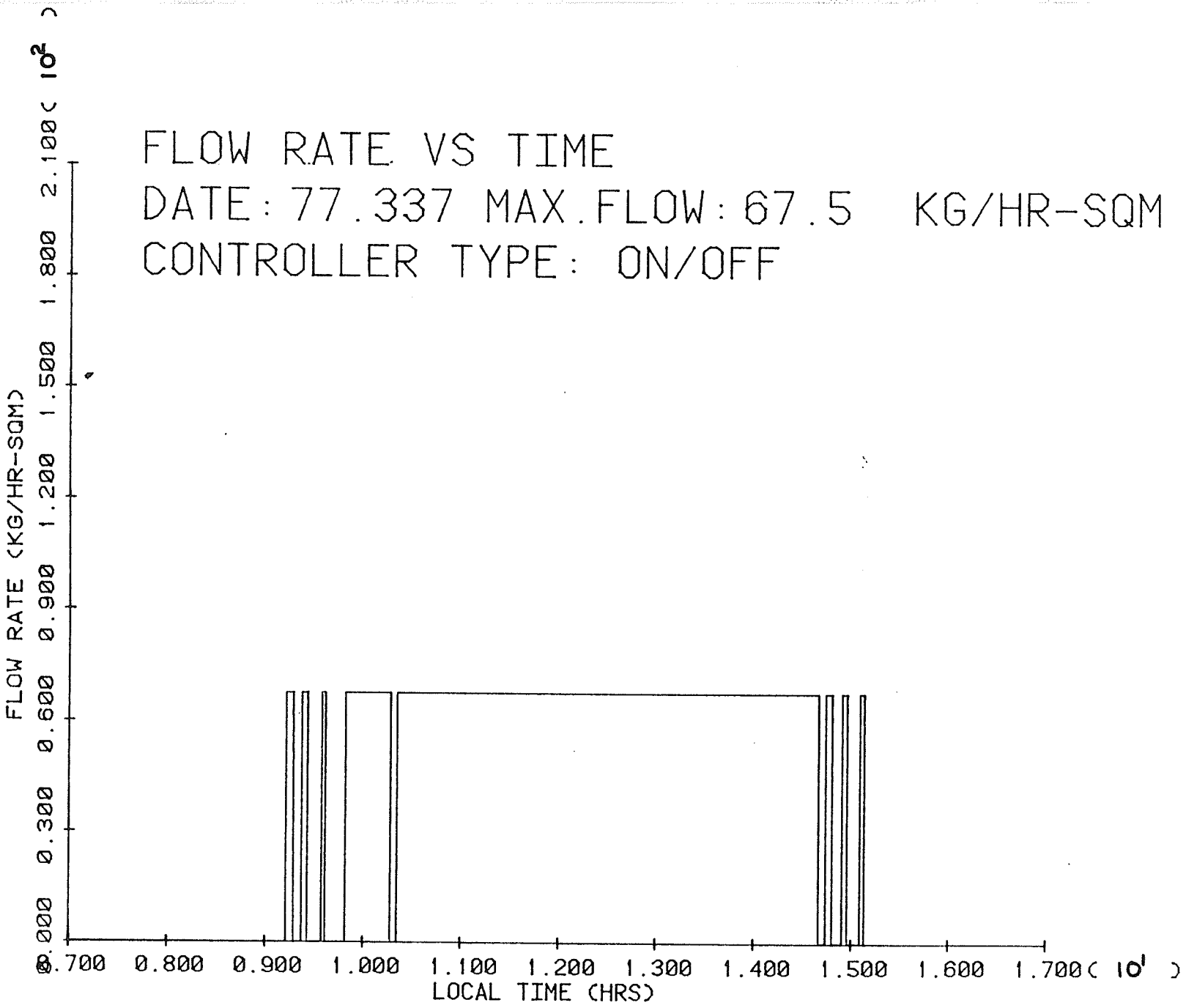

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
FLOW RATE (KG/HR-SQM)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100 (10^2)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10^1)
LOCAL TIME (HRS)

FIGURE 21

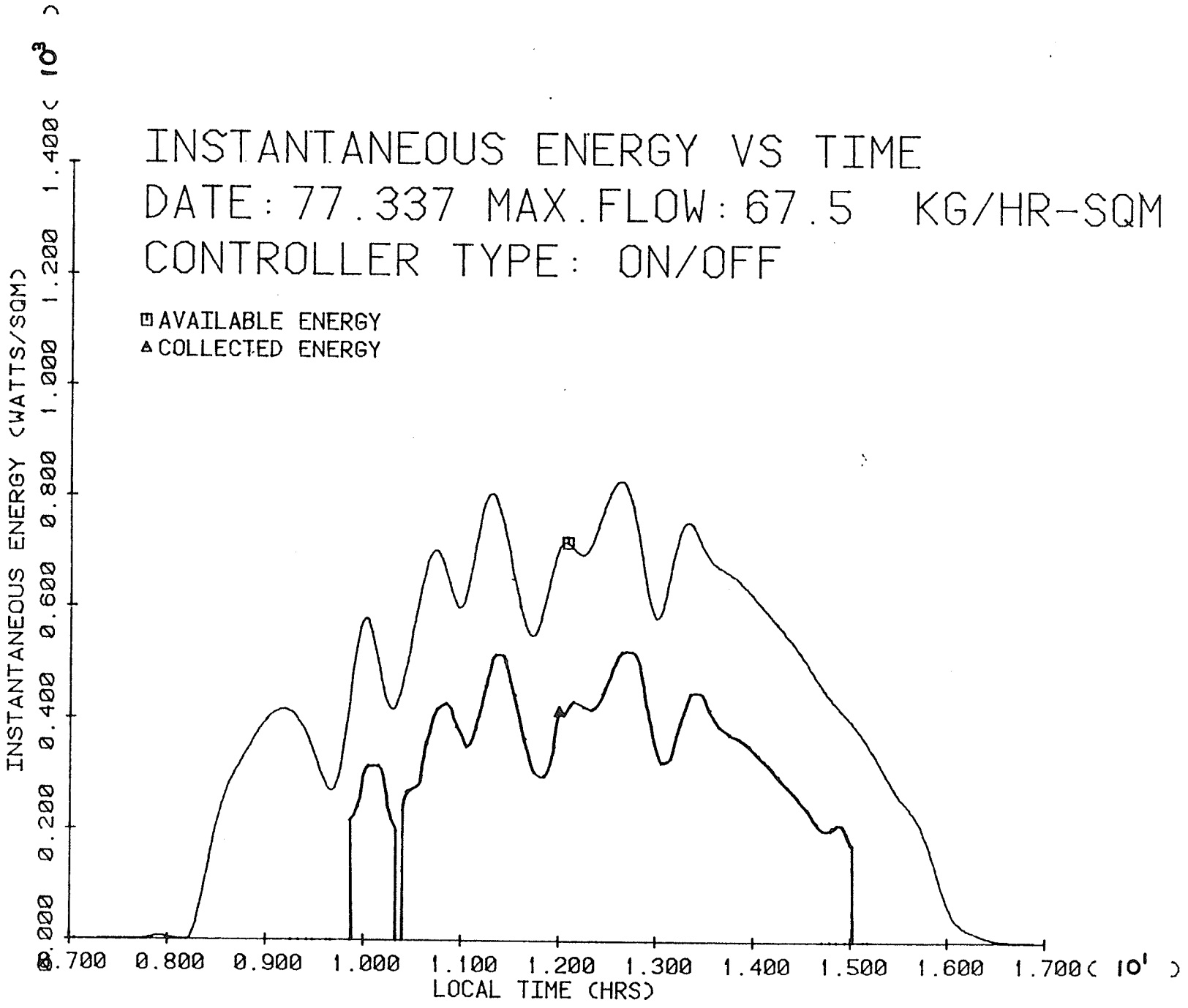

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
0.000 0.200 0.400 0.600 0.800 1.000 1.200 1.400 (10³)
0.700 0.800 0.900 1.000 1.100 1.200 1.300 1.400 1.500 1.600 1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 22

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX. FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
EFFICIENCY (PER CENT)
1.000
0.800
0.600
0.400
0.200
0.000
0.740 0.827 0.914 1.001 1.088 1.175 1.262 1.349 1.436 1.523 1.610 (10^-1)
(TAV-TAMB)/I
(10^2)

FIGURE 23

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
COLLECTOR PLATE
INNER COVER PLATE
OUTER COVER PLATE
TANK
TEMPERATURE (C)
9.000
7.000
5.000
3.000
1.000
-1.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
LOCAL TIME (HRS)

FIGURE 24

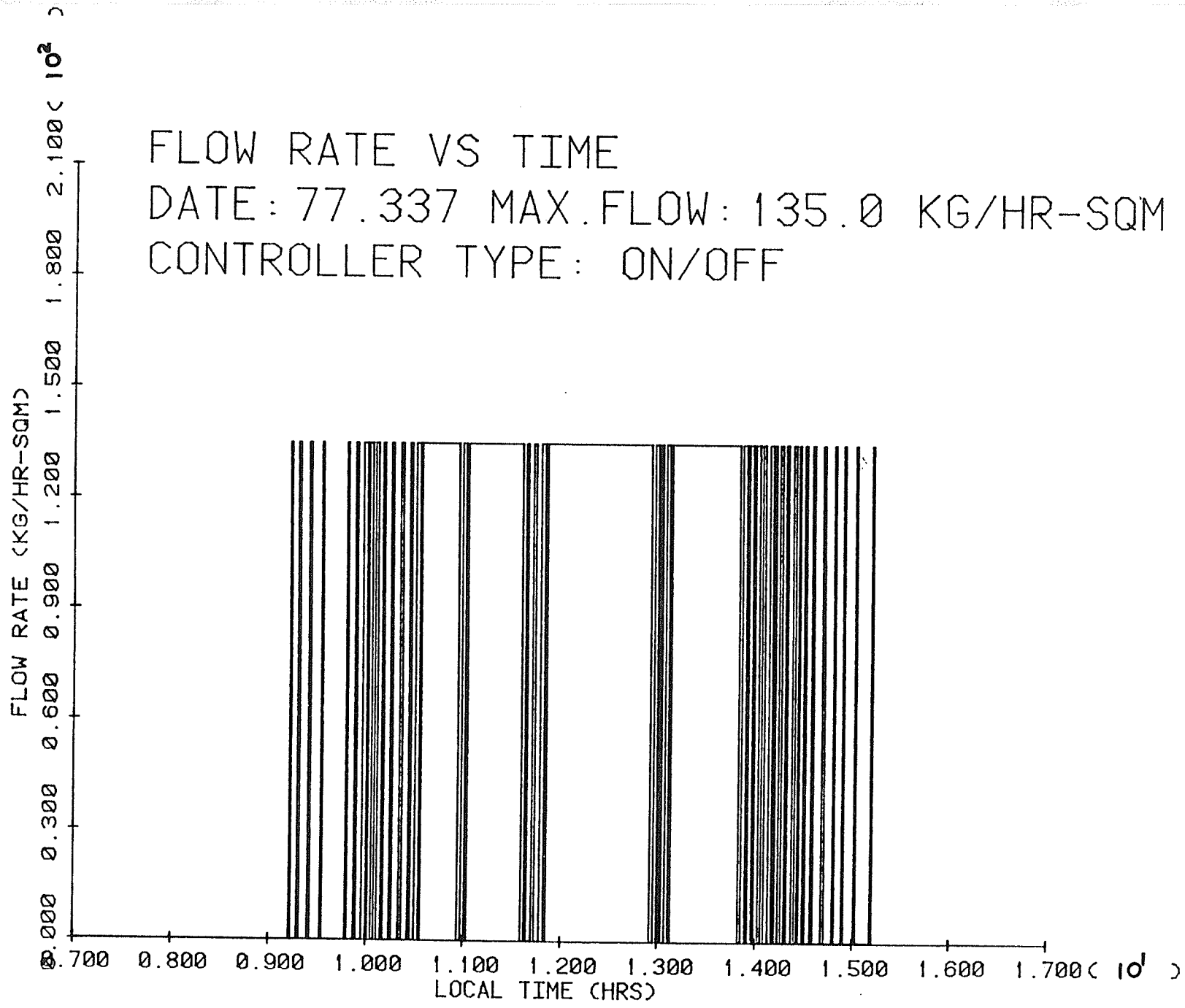

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
FLOW RATE (KG/HR-SQM)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100 (10²)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 25

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
LOCAL TIME (HRS)
0.000
0.200
0.400
0.600
0.800
1.000
1.200
1.400 (10³)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)

FIGURE 26

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
EFFICIENCY (PER CENT)
1.000
0.800
0.600
0.400
0.200
0.000
0.704
0.787
0.870
0.953
1.036
1.119
1.203
1.286
1.369
1.452
1.535
(TAV-TAMB)/I
(10^-1)

FIGURE 27

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: ON/OFF
□ COLLECTOR PLATE
△ INNER COVER PLATE
+ OUTER COVER PLATE
Y TANK
TEMPERATURE (C)
9.000 (10')
7.000
5.000
3.000
1.000
-1.000
-3.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10')
LOCAL TIME (HRS)

FIGURE 28

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
FLOW RATE (KG/HR-SQM)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100 (10²)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 29

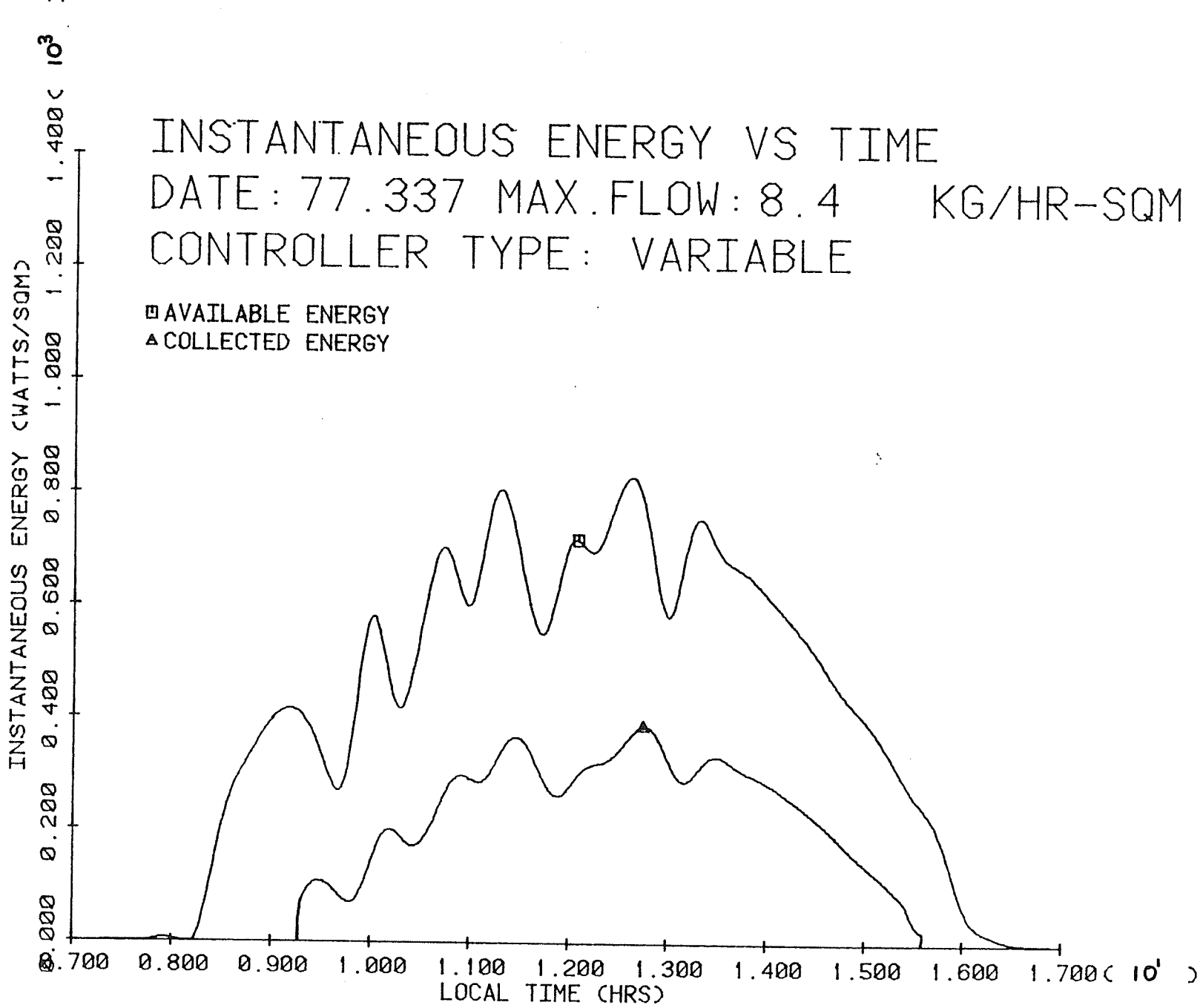

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
0.000 0.200 0.400 0.600 0.800 1.000 1.200 1.400 (10^3)
0.700 0.800 0.900 1.000 1.100 1.200 1.300 1.400 1.500 1.600 1.700 (10')
LOCAL TIME (HRS)

FIGURE 30

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX. FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
EFFICIENCY (PER CENT)
(TAV-TAMB)/I
0.000
0.200
0.400
0.600
0.800
1.000
0.979
1.133
1.287
1.440
1.594
1.748
1.902
2.056
2.209
2.363
2.517
(10^{-1})
(10^{2})

FIGURE 31

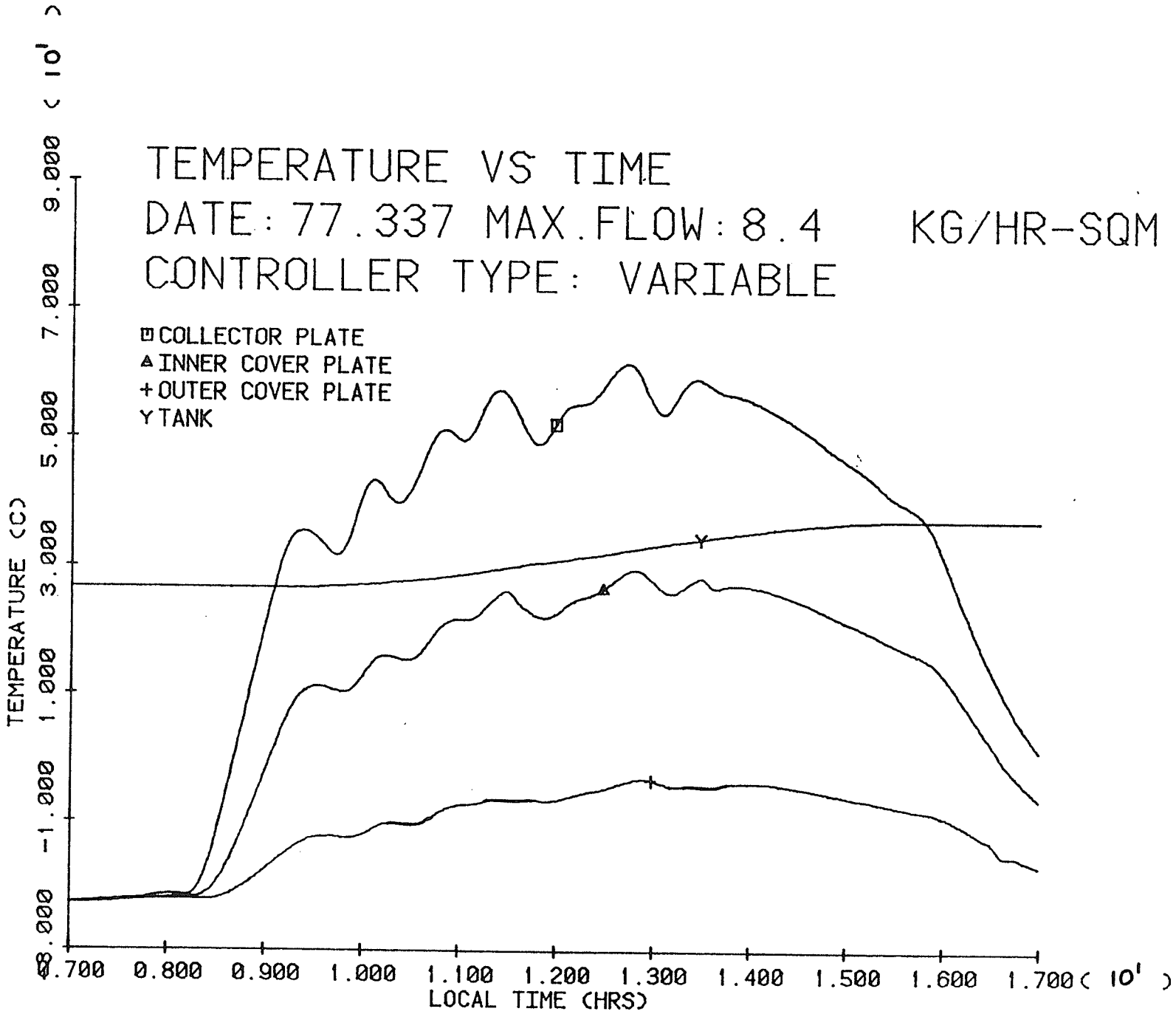

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 8.4 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
COLLECTOR PLATE
INNER COVER PLATE
OUTER COVER PLATE
TANK
TEMPERATURE (C)
-3.000
-1.000
1.000
3.000
5.000
7.000
9.000
(10')
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
(10')
LOCAL TIME (HRS)

FIGURE 32

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
FLOW RATE (KG/HR-SQM)
0.000 0.300 0.600 0.900 1.200 1.500 1.800 2.100 (10²)
0.700 0.800 0.900 1.000 1.100 1.200 1.300 1.400 1.500 1.600 1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 33

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
0.000
0.200
0.400
0.600
0.800
1.000
1.200
1.400 (10³)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 34

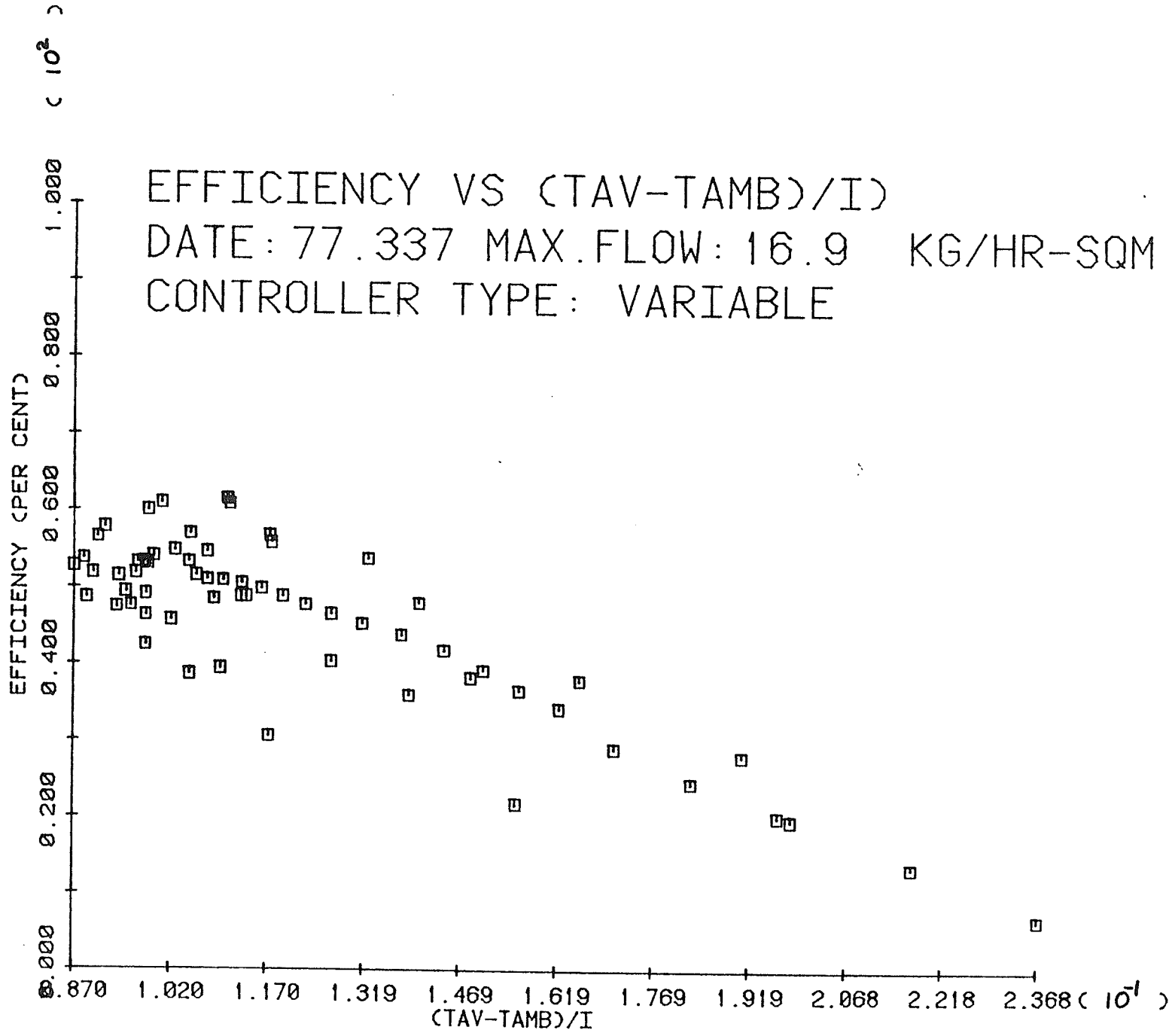

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
(10^2)
0.870
1.020
1.170
1.319
1.469
1.619
1.769
1.919
2.068
2.218
2.368 (10^-1)
(TAV-TAMB)/I

FIGURE 35

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 16.9 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
COLLECTOR PLATE
INNER COVER PLATE
OUTER COVER PLATE
TANK
TEMPERATURE (C)
9.000 (10')
7.000
5.000
3.000
1.000
-1.000
-3.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10')
LOCAL TIME (HRS)

FIGURE 36

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
FLOW RATE (KG/HR-SQM)
(10²)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
LOCAL TIME (HRS)
(10¹)

FIGURE 37

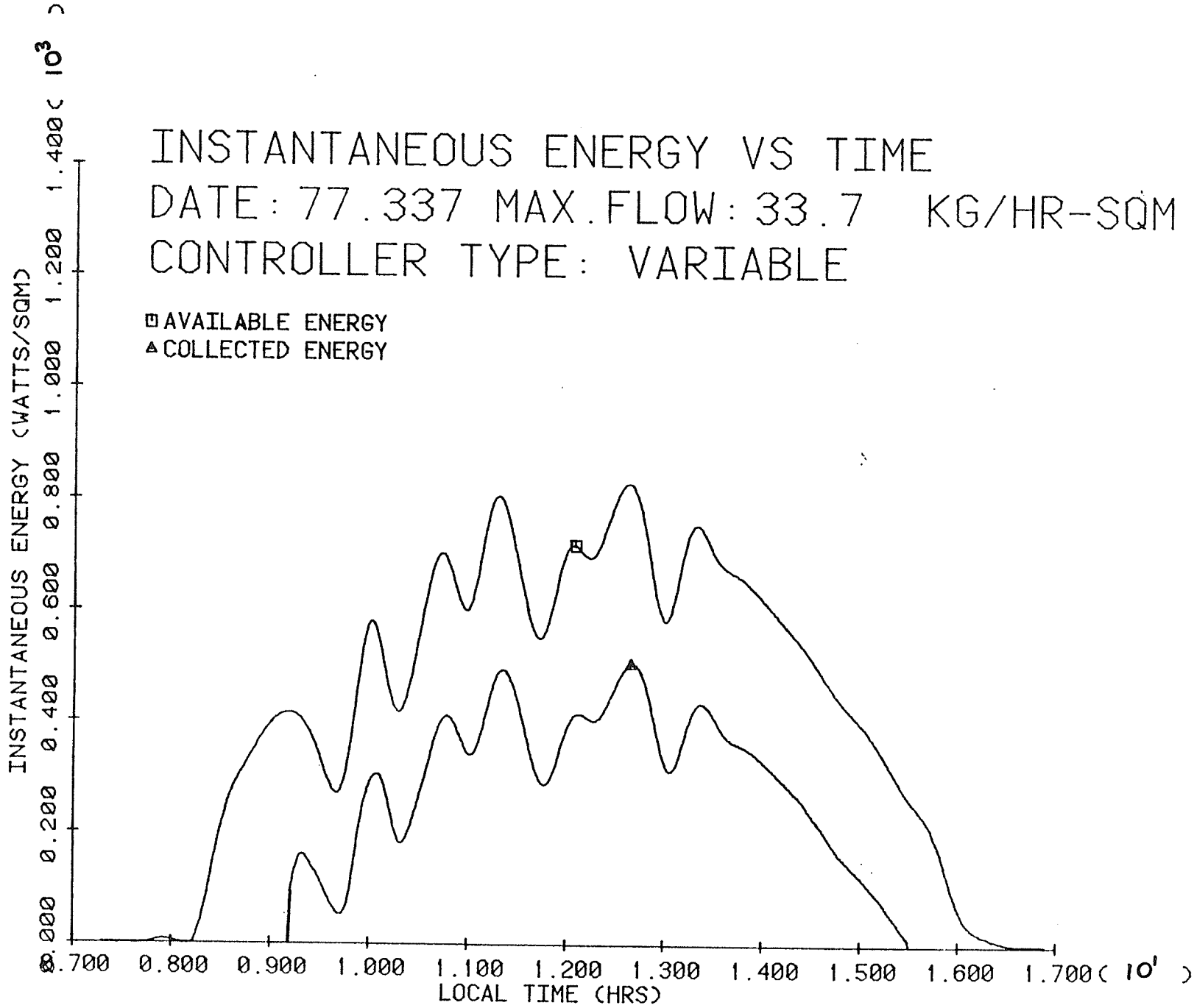

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM) (10³)
0.000
0.200
0.400
0.600
0.800
1.000
1.200
1.400
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 38

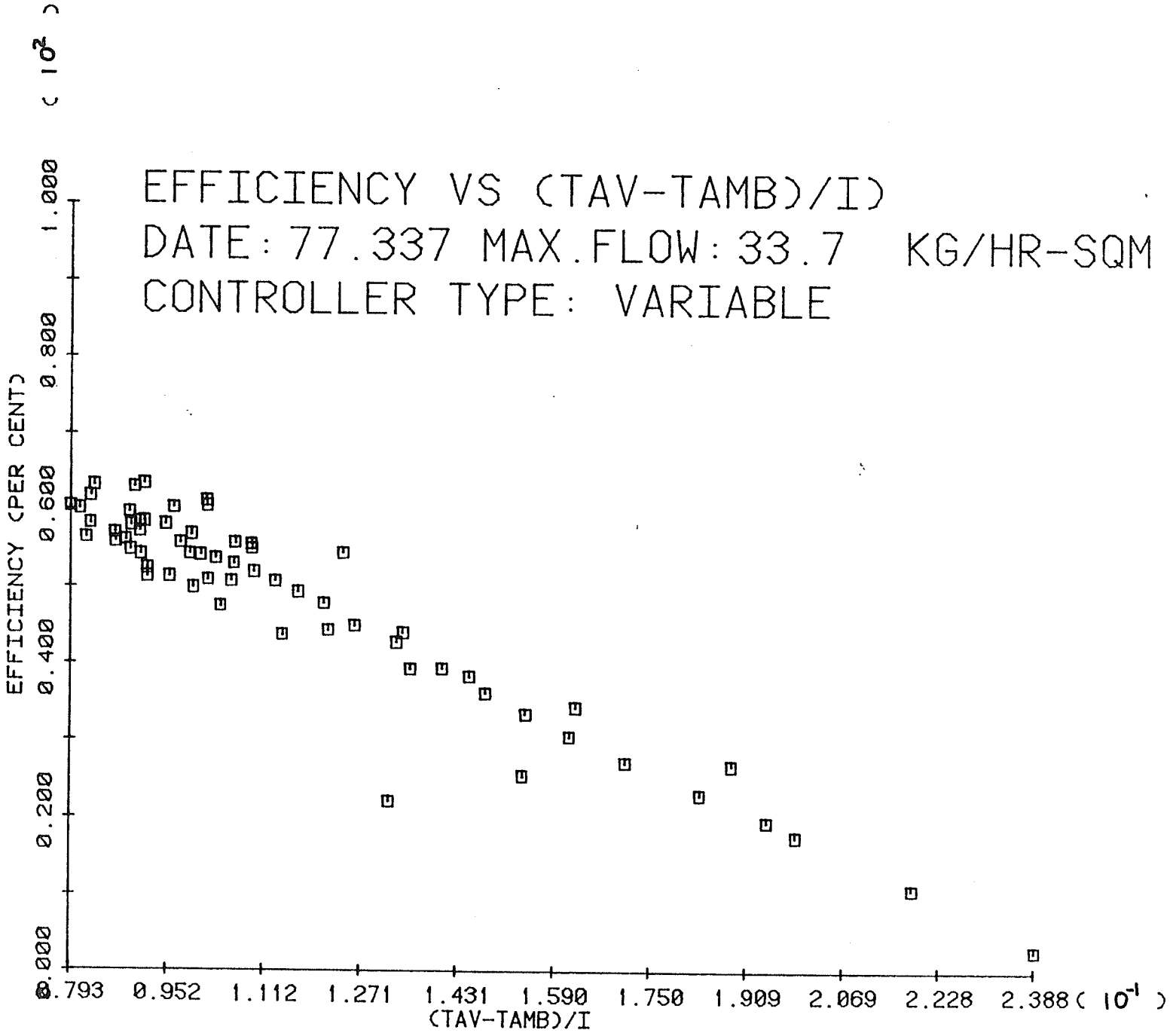

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
(10^2)
0.793
0.952
1.112
1.271
1.431
1.590
1.750
1.909
2.069
2.228
2.388 (10^-1)
(TAV-TAMB)/I

FIGURE 39

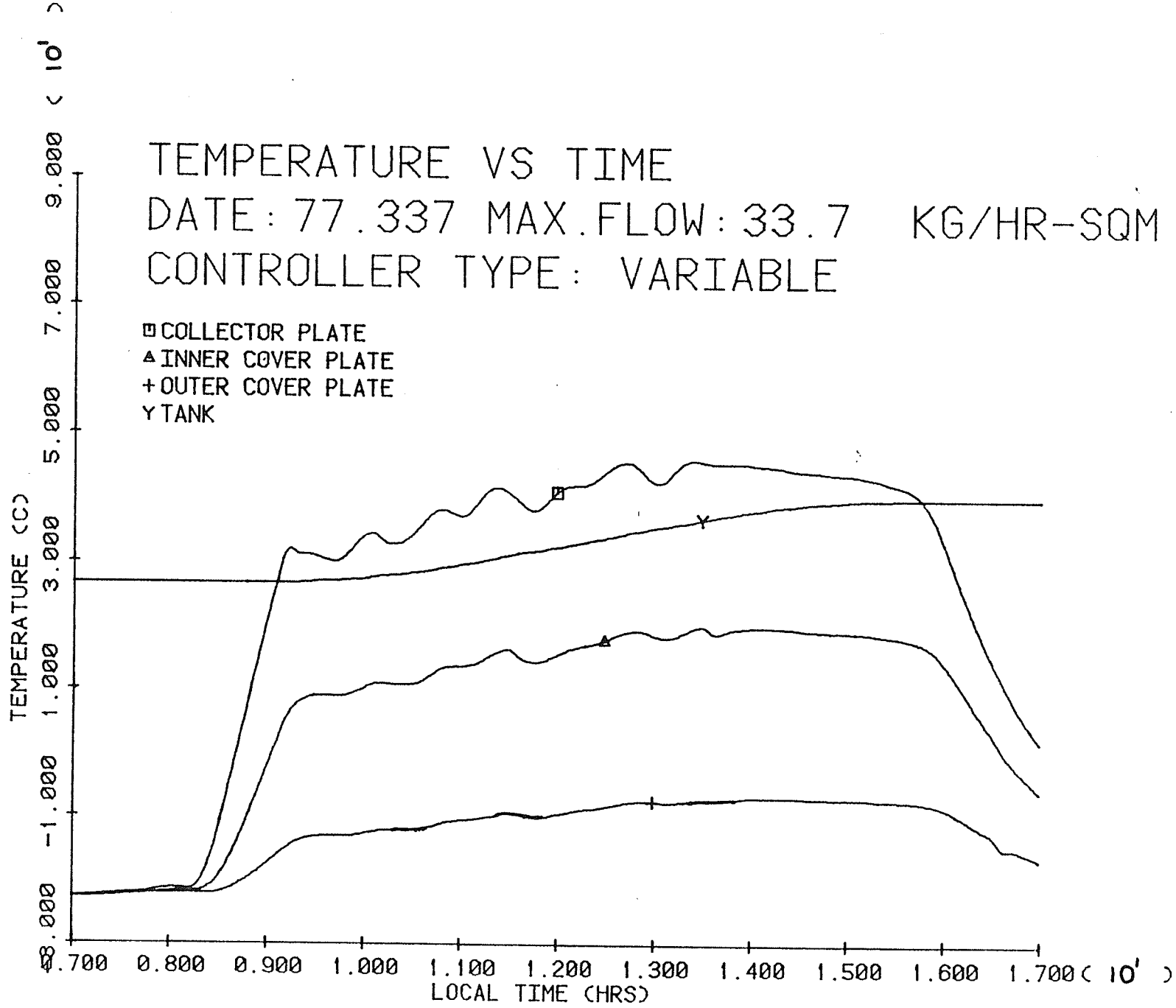

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 33.7 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
COLLECTOR PLATE
INNER COVER PLATE
OUTER COVER PLATE
TANK
TEMPERATURE (C)
-3.000
-1.000
1.000
3.000
5.000
7.000
9.000 (10')
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10')
LOCAL TIME (HRS)

FIGURE 40

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
FLOW RATE (KG/HR-SQM)
(10²)
2.100
1.800
1.500
1.200
0.900
0.600
0.300
0.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 41

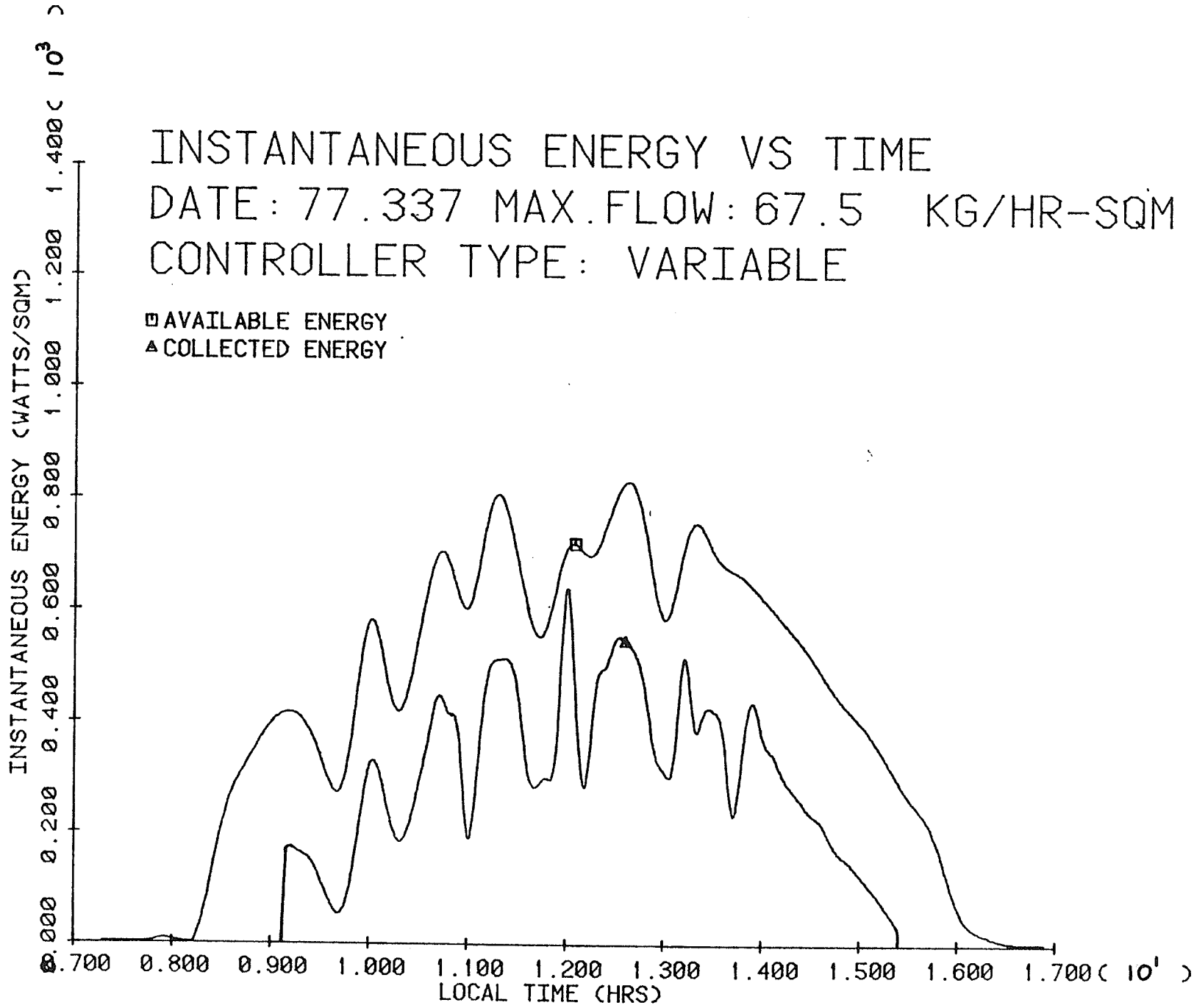

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
0.000 0.200 0.400 0.600 0.800 1.000 1.200 1.400 (10^3)
0.700 0.800 0.900 1.000 1.100 1.200 1.300 1.400 1.500 1.600 1.700 (10')
LOCAL TIME (HRS)

FIGURE 42

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
EFFICIENCY (PER CENT)
1.000
0.800
0.600
0.400
0.200
0.000
0.743 0.888 1.032 1.177 1.321 1.466 1.610 1.755 1.899 2.044 2.188 (10⁻¹)
(TAV-TAMB)/I

FIGURE 43

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 67.5 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
□ COLLECTOR PLATE
△ INNER COVER PLATE
+ OUTER COVER PLATE
Y TANK
TEMPERATURE (C)
-1.000
1.000
3.000
5.000
7.000
9.000
LOCAL TIME (HRS)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700

FIGURE 44

FLOW RATE VS TIME
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
FLOW RATE (KG/HR-SQM)
0.000
0.300
0.600
0.900
1.200
1.500
1.800
2.100 (10²)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10')
LOCAL TIME (HRS)

FIGURE 45

INSTANTANEOUS ENERGY VS TIME
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
AVAILABLE ENERGY
COLLECTED ENERGY
INSTANTANEOUS ENERGY (WATTS/SQM)
0.000
0.200
0.400
0.600
0.800
1.000
1.200
1.400 (10³)
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700 (10¹)
LOCAL TIME (HRS)

FIGURE 46

EFFICIENCY VS (TAV-TAMB)/I)
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
0.724
0.871
1.017
1.164
1.310
1.457
1.604
1.750
1.897
2.043
2.190
(TAV-TAMB)/I
(10^{-1})
(10^{2})

FIGURE 47

TEMPERATURE VS TIME
DATE: 77.337 MAX.FLOW: 135.0 KG/HR-SQM
CONTROLLER TYPE: VARIABLE
⊡ COLLECTOR PLATE
▲ INNER COVER PLATE
+ OUTER COVER PLATE
Y TANK
TEMPERATURE (C)
LOCAL TIME (HRS)
-1.000
1.000
3.000
5.000
7.000
9.000
0.700
0.800
0.900
1.000
1.100
1.200
1.300
1.400
1.500
1.600
1.700
(10')

FIGURE 48

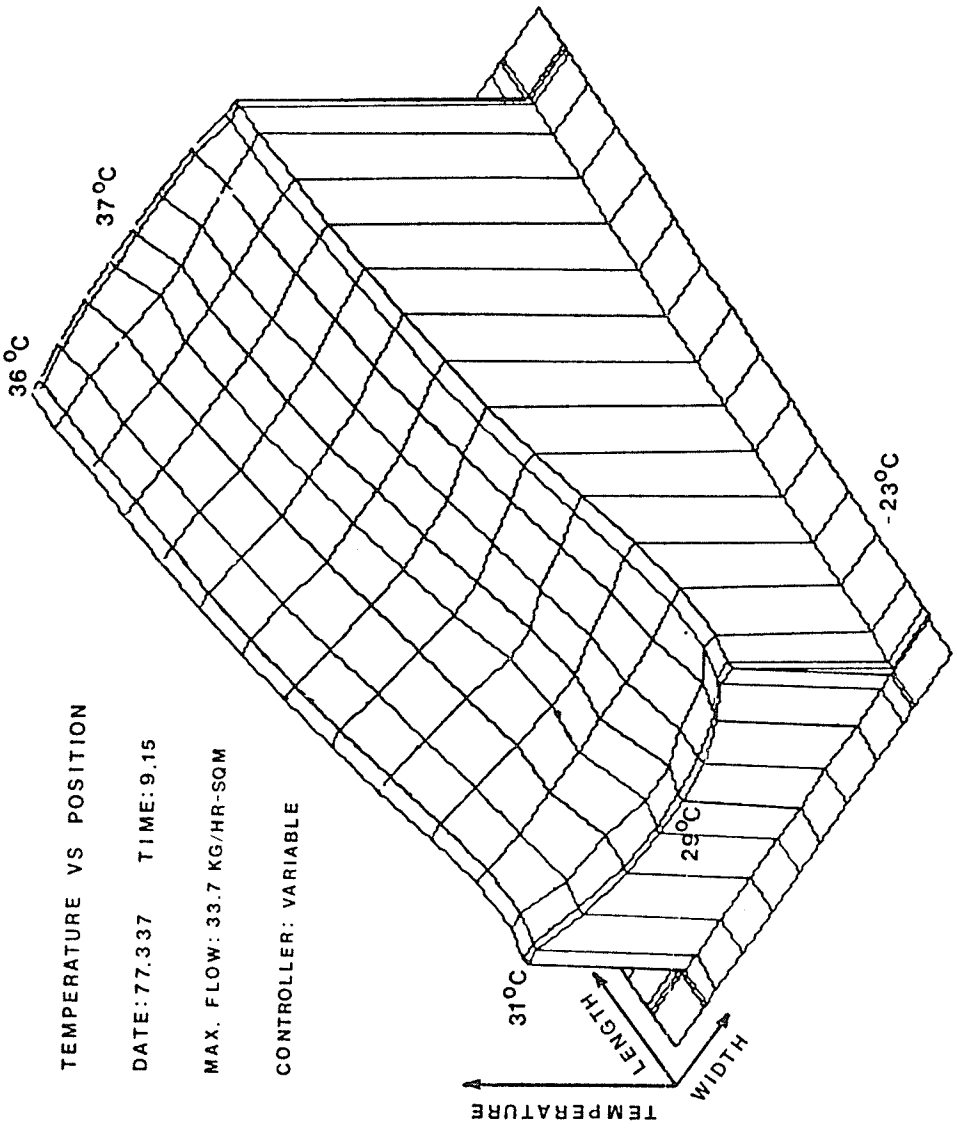

TEMPERATURE VS POSITION
DATE: 77.337 TIME: 9.15
MAX. FLOW: 33.7 KG/HR-SQM
CONTROLLER: VARIABLE
36 °C
37°C
-23°C
31°C
LENGTH
WIDTH
TEMPERATURE

FIGURE 49

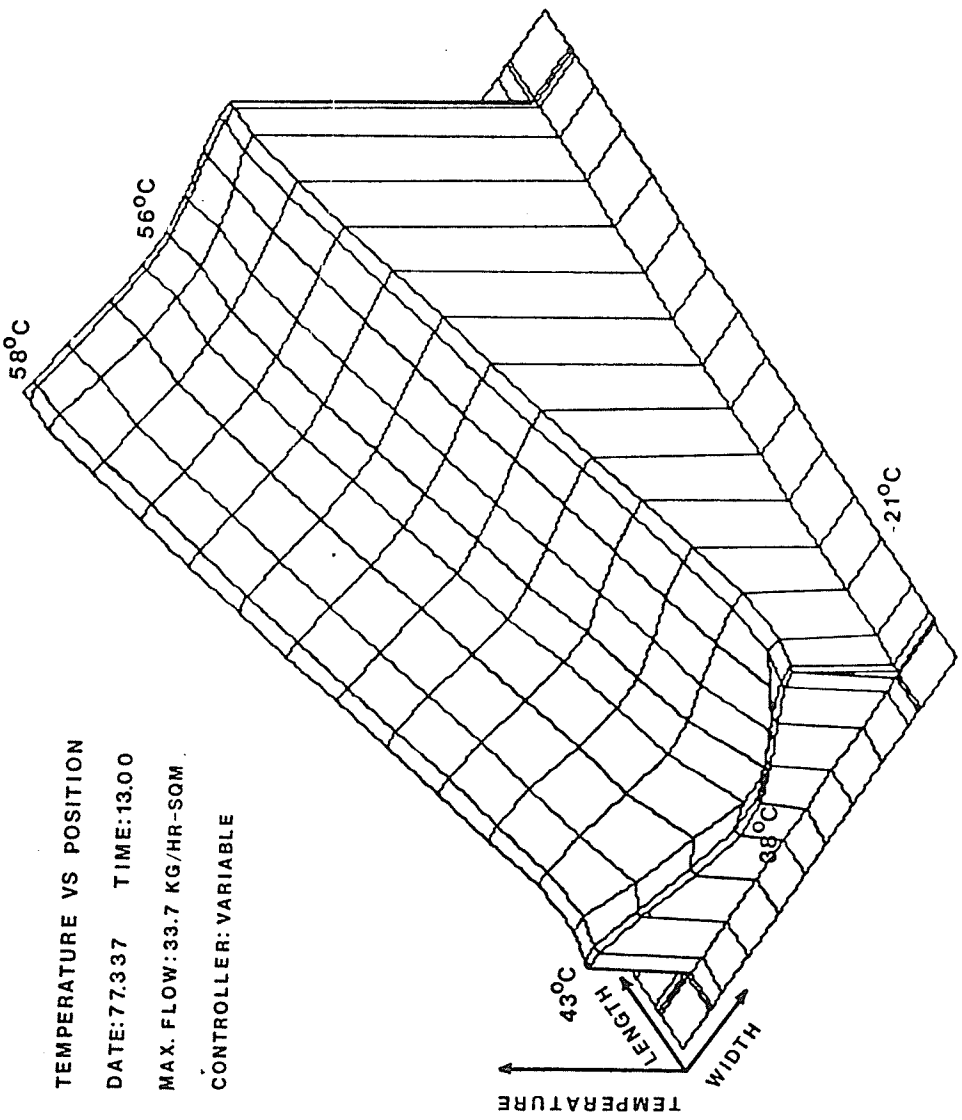

TEMPERATURE VS POSITION
DATE:77.337 TIME:13.00
MAX. FLOW:33.7 KG/HR-SQM
CONTROLLER: VARIABLE
58°C
56°C
21°C
43°C
LENGTH
WIDTH
TEMPERATURE

FIGURE 50

DAILY EFFICIENCY VS FLOW RATE
DATE: 77.231
FIXED FLOW CONTROLLER
VARIABLE FLOW CONTROLLER
DAILY EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
0.000
0.150
0.300
0.450
0.600
0.750
0.900
1.050
1.200
1.350
1.500
FLOW RATE (KG/HR-SQM)
(10^2)

FIGURE 51

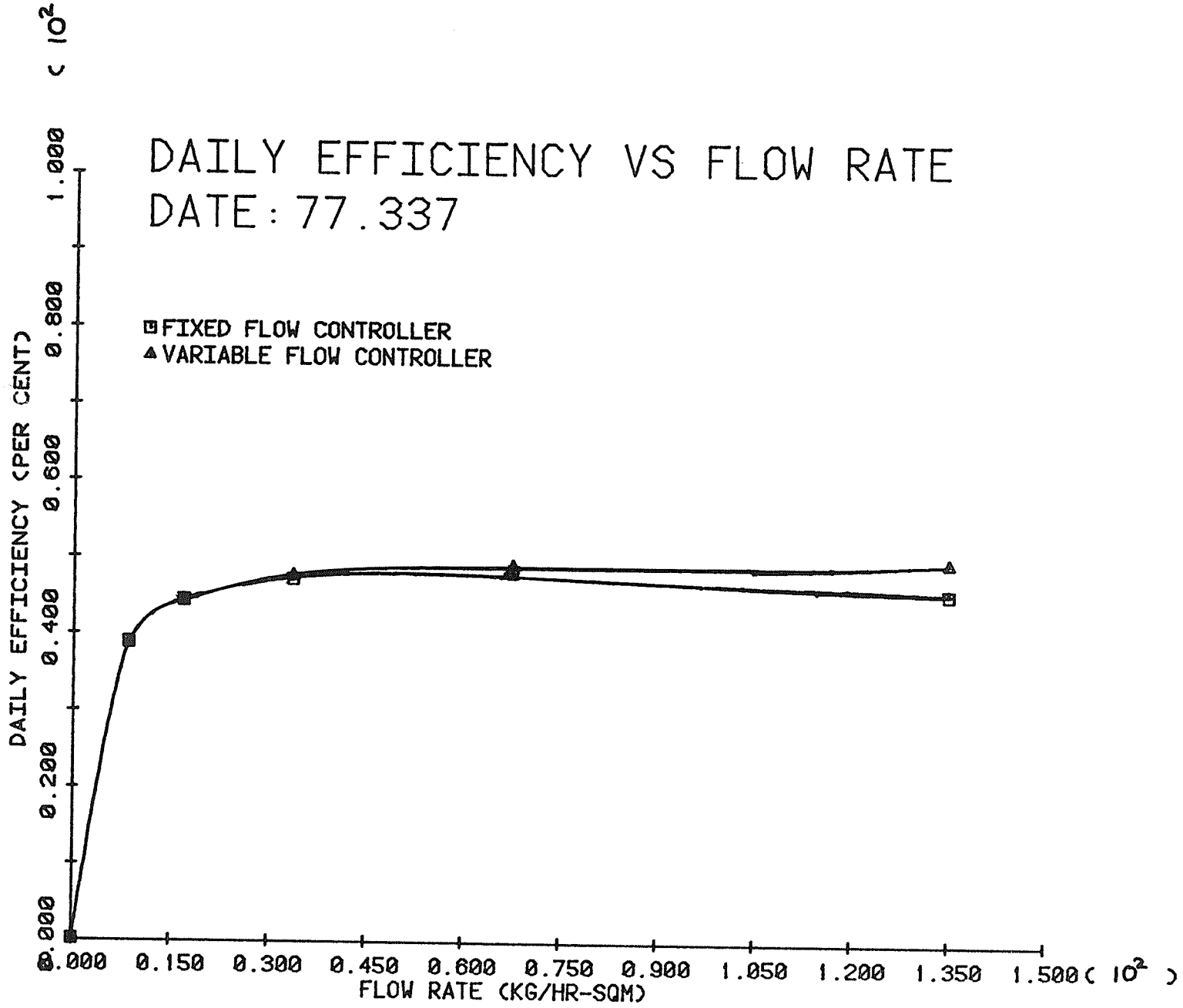

DAILY EFFICIENCY VS FLOW RATE
DATE: 77.337
FIXED FLOW CONTROLLER
VARIABLE FLOW CONTROLLER
DAILY EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
(10^2)
0.000
0.150
0.300
0.450
0.600
0.750
0.900
1.050
1.200
1.350
1.500 (10^2)
FLOW RATE (KG/HR-SQM)

FIGURE 52

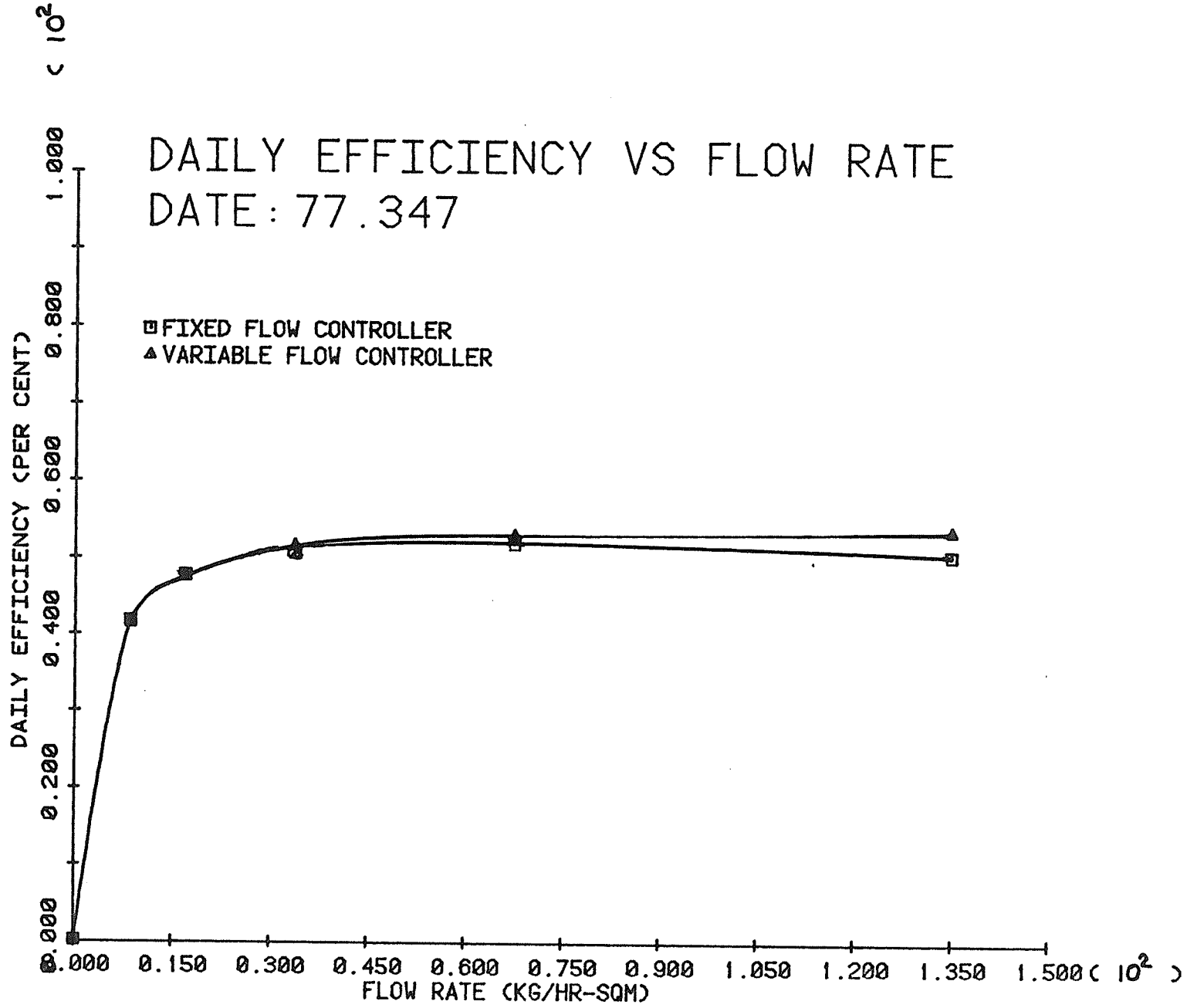

DAILY EFFICIENCY VS FLOW RATE
DATE: 77.347
FIXED FLOW CONTROLLER
VARIABLE FLOW CONTROLLER
DAILY EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
(10^2)
0.000
0.150
0.300
0.450
0.600
0.750
0.900
1.050
1.200
1.350
1.500
(10^2)
FLOW RATE (KG/HR-SQM)

FIGURE 53

DAILY EFFICIENCY VS FLOW RATE
DATE: 77.360
FIXED FLOW CONTROLLER
VARIABLE FLOW CONTROLLER
DAILY EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
0.000
0.150
0.300
0.450
0.600
0.750
0.900
1.050
1.200
1.350
1.500
FLOW RATE (KG/HR-SQM)
(10²)

FIGURE 54

DAILY EFFICIENCY VS FLOW RATE
DATE: 78.001
FIXED FLOW CONTROLLER
VARIABLE FLOW CONTROLLER
DAILY EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
(10^2)
0.000
0.150
0.300
0.450
0.600
0.750
0.900
1.050
1.200
1.350
1.500 (10^2)
FLOW RATE (KG/HR-SQM)

FIGURE 55

DAILY EFFICIENCY VS FLOW RATE
DATE: 78.002
FIXED FLOW CONTROLLER
VARIABLE FLOW CONTROLLER
DAILY EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
0.000
0.150
0.300
0.450
0.600
0.750
0.900
1.050
1.200
1.350
1.500
FLOW RATE (KG/HR-SQM)
(10²)

FIGURE 56

DAILY EFFICIENCY VS FLOW RATE
DATE: 78.007
FIXED FLOW CONTROLLER
VARIABLE FLOW CONTROLLER
DAILY EFFICIENCY (PER CENT)
0.000
0.200
0.400
0.600
0.800
1.000
(10²)
0.000
0.150
0.300
0.450
0.600
0.750
0.900
1.050
1.200
1.350
1.500
FLOW RATE (KG/HR-SQM)
(10²)

FIGURE 57

/46

APPENDIX I

FINITE DIFFERENCE PROGRAM LISTING

This appendix contains Fortran listings of the finite difference program. The main program is presented first followed by listings of all of the subroutines required (except for subroutine SEARCH which is presented in Z-80 Assembler language in Appendix II).

Most of the subroutines were designed, debugged, and then stored as relocatable object modules or libraries in two separate files: THRMLLIB.REL and FTNUTIL.FOR. This procedure allowed linking the main program with the necessary subroutines without having to re-compile each subroutine.

The program and subroutines are well documented and self explanatory. Note that some of the subroutines (e.g., TPRINT) are not called in the version of the program that has been listed, however, these subroutines were used heavily in the development of the program and are also useful for normal runs.

```
      PROGRAM THERML
C
C PURPOSE:       THIS IS THE MAIN PROGRAM FOR THE THERMAL ANALYSES
C                OF A FLAT PLATE WATER COOLED SOLAR ENERGY COLLECTOR
C                USING FINITE DIFFERENCES.  INPUTS, OUTPUTS, AND
C                RESTRICTIONS ARE TOO NUMEROUS TO REPEAT HERE.  THE
C                READER SHOULD REFER TO THE THESIS TEXT AND THE TEXT
C                AT THE FRONT OF EACH ROUTINE IN ORDER TO LEARN HOW
C                TO USE THIS PROGRAM.
C
C INPUTS:        ALL INPUT INFROMATION COMES FROM FILES RUNDATA.DAT,
C                CLCTRDAT.DAT, AND BYYDDD.DAT WHERE YYDDD IS THE YEAR
C                AND DAY OF THE BOUNDARY DATA.
C
C
C OUTPUTS:       CURRENTLY SET UP TO OUTPUT TEMPERATURE DATA ONTO
C                TYYDDD.DAT WHERE YYDDD IS THE YEAR AND DAY BEING
C                ANALYZED.  FFOR ADDITIONAL DATA SEE THESIS TEXT.
C
C RESTRICTIONS: USES ENGLISH ENGINEERING UNITS.  NOTE THAT THE PRINT-
C                OUT OF THE DATA MAY CONVERTED INTO OTHER UNITS IN THE
C                SUBROUTINE PRNOUT.
C
C
C SUBROUTINES REFERENCED:        UNPACK,GETINF,CONVRS,DCSINT,STRADK,
C                                SOLVE.
C
C START
C
      LOGICAL TFLNM(11),QFLNM(11),GFLNM(11),PFLNM(11),BFLNM(11)
      LOGICAL E,FIRST,IBUF(128),CTRLLR
      COMMON/TEMPO/TOLD(130)
      COMMON/TEMPN/TNEW(130)
      COMMON/CAP/C(130)
      COMMON/HEAT/Q(130)
      COMMON/ARRAYS/A(130)
```

```
      COMMON/NODEA/NA(420)
      COMMON/NODEB/NB(420)
      COMMON/COND/G(420)
      COMMON/MODEL/ANAME(9)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
      COMMON/BNDRY/NDATA1,NDATA2,TIMHR1(50),TIMHR2(50),SOLAR(50),
     &             TMPAMB(50),WNDS(50),WNDD(50)
      COMMON/COEFF/SOLARB(50),SOLARC(50),SOLARD(50),
     &             TAMBB(50),TAMBC(50),TAMBD(50),
     &             WNDSB(50),WNDSC(50),WNDSD(50),
     &             WNDDB(50),WNDDC(50),WNDDD(50)
      COMMON/TSTVAL/ITRSUM,FLRATE,IFRST,FLORAT,DELTAT,CTRLLR
      DATA TFLNM/'T',7*' ','D','A','T'/,QFLNM/'Q'/,GFLNM/'G'/,
     &     PFLNM/'P'/,E/1HE/,FIRST/.TRUE./,
     &     BFLNM/'B',7*' ','D','A','T'/
C
C READ IN RUN PARAMETERS FROM RUNDATA.DAT
C
      IFRST=0
      IREC=0
      CALL OPEN(9,'RUNDATA DAT',1)
C
10    IF(IREC.EQ.0)GO TO 20
      IFRST=0
      ITRSUM=0
      REWIND 9
      FIRST=.TRUE.
C
      DO 30 I=1,IREC
30    CALL UNPACK(9,IBUF,FIRST)
C
20    CALL UNPACK(9,IBUF,FIRST)
      DECODE(IBUF,1000)(BFLNM(I),I=2,6)
1000  FORMAT(5A1)
```

```
C
      IF(BFLNM(2),EQ,E)GO TO 9999
C
      DECODE(IBUF,1010)(TFLNM(I),I=2,8)
1010  FORMAT(8A1)
C
      DECODE(IBUF,1020)TIMEN,TIMEND,FLRATE,IDRIVE
1020  FORMAT(8X,3G14.7,I2)
C
      DECODE(IBUF,1030)CTRLLR
1030  FORMAT(5X,A1)
C
      WRITE(1,1040)(TFLNM(I),I=1,11),TIMEN,TIMEND,FLRATE,CTRLLR,
     &  IDRIVE
1040  FORMAT(1H ,11A1,3(5X,F6.2),5X,A1,5X,I2)
C
      DO 40 I=2,11
      QFLNM(I)=TFLNM(I)
      GFLNM(I)=TFLNM(I)
40    PFLNM(I)=TFLNM(I)
C
      TIMEO=TIMEN
      IREC=IREC+1
C
      CALL OPEN(6,'CLCTRDATDAT',IDRIVE)
      CALL OPEN(7,BFLNM,IDRIVE)
C
C LOAD ALL ARRAYS WITH THE INPUT DATA
C
      CALL GETINP(6,7)
C
C ALL INPUT DATA READ IN  THEREFORE CLOSE INPUT FILES.
C
      ENDFILE 6
      ENDFILE 7
C
```

```
C CONVERT BOUNDARY DATA TO ENGLISH ENGINEERING UNITS
C
      CALL CONVRS
C
C CALCULATE THE INTERPOLATION COEFFICIENTS FOR THE BOUNDARY DATA
C
      CALL DCSINT(2,0.01,NDATA1,TIMHR1,SOLAR,0.0,0.0,SOLARB,SOLARC,
     &            SOLARD)
      CALL DCSINT(2,0.01,NDATA2,TIMHR2,TMPAMB,0.0,0.0,TAMBB,TAMBC,
     &            TAMBD)
      CALL DCSINT(2,0.01,NDATA2,TIMHR2,WNDS,0.0,0.0,WNDSB,WNDSC,WNDSD)
      CALL DCSINT(2,0.01,NDATA2,TIMHR2,WNDD,0.0,0.0,WNDDB,WNDDC,WNDDD)
C
C
C OPEN THE OUTPUT FILES
C
      CALL OPEN(7,TFLNM,IDRIVE)
C
C CONVERT ALL AREAS IN A( ) TO SQ.FT.
C
      DO 50 I=1,NNOD
50    A(I)=A(I)/144.0
C
C SET ALL RADIATION CONDUCTORS
C
      CALL STRADK
C
C SET THE INITIAL TEMPERATURES TO AMBIENT
C
      TNEW(123)=TMPAMB(1)
      DO 60 I=1,NNOD
60    TNEW(I)=TNEW(123)
C
C SET TANK TEMPERATURE TO 80.0 F.
C
      TNEW(120)=80.0
```

```
C
      CALL SOLVE
C
C CLOSE ALL FILES
C
      ENDFILE 7
C
      GO TO 10
9999  ENDFILE 9
C
      STOP
      END
C
      SUBROUTINE PRE
C
C PURPOSE:      THIS SUBROUTINE IS CALLED BY SUBROUTINE SOLVE PRIOR
C               TO SOLVING THE SET OF EQUATIONS AT EACH TIME STEP.
C               BOUNDARY DATA SUCH AS AMBIENT TEMPERATURE, SOLAR
C               LOADING, WIND SPEED, ETC. IS SET IN THIS ROUTINE
C               SO THAT SOLVE IS PRESENTED WITH A NEW SET OF
C               CONDITIONS.  ALSO THE CONTROLLER LOGIC IS HANDLED
C               IN THIS ROUTINE.  MASS FLOW CONDUCTORS ARE SET HERE
C               DEPENDING ON THE DECISION OF THE CONTROLLER LOGIC.
C
C INPUTS:       ALL INPUTS ARE CONTAINED WITHIN THE COMMON BLOCKS.
C
C OUTPUTS:      ALL OUTPUTS ARE CONTAINED WITHIN THE COMMON BLOCKS.
C
C SUBROUTINES REFERENCED:       STFLOW,PRNOUT,STTUBK,SPLEVL,STPLHC,
C                               STFLUX
C
C RESTRICTIONS: NONE.
C
C START
C
      LOGICAL PUMP,PMPSTA,CTRLLR,V
```

```
      COMMON/TEMPN/TNEW(130)
      COMMON/CAP/C(130)
      COMMON/HEAT/Q(130)
      COMMON/ARRAYS/A(130)
      COMMON/NODEA/NA(420)
      COMMON/NODEB/NB(420)
      COMMON/COND/G(420)
      COMMON/MODEL/ANAME(9)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
      COMMON/BNDRY/NDATA1,NDATA2,TIMHR1(50),TIMHR2(50),SOLAR(50),
     &             TMPAMB(50),WNDS(50),WNDD(50)
      COMMON/COEFF/SOLARB(50),SOLARC(50),SOLARD(50),
     &             TAMBB(50),TAMBC(50),TAMBD(50),
     &             WNDSB(50),WNDSC(50),WNDSD(50),
     &             WNDDB(50),WNDDC(50),WNDDD(50)
      COMMON/TSTVAL/ITRSUM,FLRATE,IFRST,FLORAT,DELTAT,CTRLLR
      DATA PUMP,PMPSTA/.FALSE.,.TRUE./
      DATA V/1HV/
C
      DELTAT=TNEW(52)-TNEW(120)
C
      IF(CTRLLR.EQ.V)GO TO 500
C
      IF(IFRST.EQ.1)GO TO 10
      PUMP=.FALSE.
      PMPSTA=.TRUE.
10    CONTINUE
C
C ***************** START OF ON/OFF PUMP LOGIC ************************
C
C
C PUMP ON AT 7.0 C (12.6 F), OFF AT 3.0C (5.4 F)
C
      IF(DELTAT.LT.5.4)GO TO 20
```

```
          IF(DELTAT.LT.12.6)GO TO 30
C
          PUMP=.TRUE.
          FLORAT=FLRATE
C
          DTIMEI=16.66667E-03
          IF(FLRATE.GT.125.0)DTIMEI=8.333333E-03
          IF(FLRATE.GE.250.0)DTIMEI=2.777777E-03
C
          GO TO 30
C
20        PUMP=.FALSE.
          FLORAT=0.0
C
          DTIMEI=33.33333E-03
          IF(DELTAT.GT.0.0)DTIMEI=8.333333E-03
          IF((DELTAT.GT.0.0).AND.(FLRATE.GE.125.0))DTIMEI=2.777777E-03
          IF((DELTAT.GT.0.0).AND.(FLRATE.GE.250.0))DTIMEI=2.083333E-03
C
30        CONTINUE
C
C IF PUMP STATUS HAS CHANGED THEN WRITE OUT TIME AND STATUS
C
          IF(PUMP.EQ.PMPSTA)GO TO 40
C
          CALL STFLOW(FLORAT)
          CALL PRNOUT
          PMPSTA=PUMP
C
40        CONTINUE
C
          GO TO 600
C
C ***************** END OF ON/OFF PUMP LOGIC ****************************
C
500       CONTINUE
```

```
C
C ***************** START OF VARIABLE FLOW RATE LOGIC ******************
C
C
C FLORAT=0.0 FOR DELTAT .LE. 3.0 C (5.4 F)
C FLORAT VARIES LINEARLY FOR 3.0 C .LE. DELTAT .LE. 7.0 C
C FLORAT=FLRATE FOR DELTAT .GE. 7.0 C (12.6 F)
C
C CALCULATE FLOW RATE
C
        FLORAT=(FLRATE/7.2)*(DELTAT-5.4)
        IF(DELTAT.LT.5.4)FLORAT=0.0
        IF(DELTAT.GE.12.6)FLORAT=FLRATE
C
C ADJUST DTIMEI
C
        IF(DELTAT.GT.0.0)DTIMEI=8.333333E-03
        IF(FLORAT.GT.125.0)DTIMEI=8.333333E-03
        IF(FLORAT.GT.250.0)DTIMEI=2.777777E-03
        CALL STFLOW(FLORAT)
C
C ***************** END OF VARIABLE FLOW RATE LOGIC ********************
C
600     IFRST=1
C
C CALCULATE TUBE CONDUCTANCES
C
        CALL STTUBK(FLORAT)
C
C SET THE AMBIENT TEMPERATURE
C
        INVL=0
        TNEW(123)=SPLEVL(TIMHR2,TMPAMB,TAMBB,TAMBC,TAMBD,
     &                   NDATA2,INVL,TIMEN)
C
C SET THE ATTIC AND SKY TEMPERATURES BASED ON THE AMBIENT
```

```
C
      TNEW(122)=TNEW(123)+20.0
      TNEW(124)=TNEW(123)-11.0
C
C SET THE PLATE TO PLATE AND PLATE TO AMBIENT CONDUCTORS
C
      INVL=0
      WNDSPD=SPLEVL(TIMHR2,WNDS,WNDSB,WNDSC,WNDSD,NDATA2,INVL,TIMEN)
      INVL=0
      WNDDIR=SPLEVL(TIMHR2,WNDD,WNDDB,WNDDC,WNDDD,NDATA2,INVL,TIMEN)
C
C SET WIND SPEED TO A LOW VALUE IF DIRECTION IS FROM
C THE NORTH I.E. THIS IS A SOUTH FACING COLLECTOR
C
      IF((WNDDIR.LT.1.570).OR.(WNDDIR.GT.4.712))WNDSPD=1.0
      CALL STPLHC(WNDSPD)
C
C DETERMINE FLUX
C
      INVL=0
      FLUX=SPLEVL(TIMHR1,SOLAR,SOLARB,SOLARC,SOLARD,NDATA1,INVL,TIMEN)
      CALL STFLUX(FLUX)
C
      RETURN
      END
C
      SUBROUTINE POST
C
C PURPOSE:      THIS SUBROUTINE IS CALLED BY SUBROUTINE SOLVE AT THE
C               COMPLETION OF SOLVING FOR THE NEW TEMPERATURES BUT
C               PRIOR TO CALLING PRNOUT.  IT'S MAIN PURPOSE IS TO
C               ALLOW SETTING UP PARAMETERS PRIOR TO ENTERING PRNOUT.
C
C INPUTS:       ALL INPUTS ARE CONTAINED WITHIN THE COMMON BLOCKS.
C
C OUTPUTS:      ALL OUTPUTS ARE CONTAINED WITHIN THE COMMON BLOCKS.
```

```
C
C SUBROUTINES REFERENCED:        SPLEVL,STFLUX
C
C RESTRICTIONS: NONE.
C
C START
C
      LOGICAL CTRLLR
      COMMON/TEMPN/TNEW(130)
      COMMON/CAP/C(130)
      COMMON/HEAT/Q(130)
      COMMON/ARRAYS/A(130)
      COMMON/NODEA/NA(420)
      COMMON/NODEB/NB(420)
      COMMON/COND/G(420)
      COMMON/MODEL/ANAME(9)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
      COMMON/BNDRY/NDATA1,NDATA2,TIMHR1(50),TIMHR2(50),SOLAR(50),
     &             TMPAMB(50),WNDS(50),WNDD(50)
      COMMON/COEFF/SOLARB(50),SOLARC(50),SOLARD(50),
     &             TAMBB(50),TAMBC(50),TAMBD(50),
     &             WNDSB(50),WNDSC(50),WNDSD(50),
     &             WNDDB(50),WNDDC(50),WNDDD(50)
      COMMON/TSTVAL/ITRSUM,FLRATE,IFRST,FLORAT,DELTAT,CTRLLR
      ITRSUM=ITRSUM+LOOPCT
C
C RESTORE Q VALUES FOR HEATING RATE PRINTS
C
      INVL=0
      FLUX=SPLEVL(TIMHR1,SOLAR,SOLARB,SOLARC,SOLARD,NDATA1,
     &            INVL,TIMEN)
      CALL STFLUX(FLUX)
C
      RETURN
```

```
      END
C
      SUBROUTINE PRNOUT
C
C PURPOSE:       THIS SUBROUTINE PROVIDES A LINK TO THE THERMAL ANALYZER
C                PRINT ROUTINES.  ALSO, THE USER MAY MODIFY VALUES AND
C                PRINT THEM OUT USING HIS OWN PRINT ROUTINES.
C                CALLED EVERY PRNOUT INTERVAL.
C
C INPUTS:        ALL INPUT INFORMATION IS PROVIDED IN THE COMMON BLOCKS.
C
C OUTPUTS:       ALL OUTPUTS HANDLED BY THE THERMAL ANALYZER ROUTINES OR
C                THE USERS ROUTINES.
C
C SUBROUTINES REFERENCED:       THE FOLLOWING ROUTINES MAY BE REFERENCED
C                IN THIS ROUTINE: HPRINT,TPRINT,QPRINT,CPRINT,
C                GPRINT,AND/OR ANY ADDITIONAL ROUTINES DEFINED
C                BY THE USER.
C
C RESTRICTIONS: NONE.
C
C START
C
      LOGICAL TFLG,CTRLLR
      COMMON/TEMPO/TOLD(130)
      COMMON/TEMPN/TNEW(130)
      COMMON/CAP/C(130)
      COMMON/HEAT/Q(130)
      COMMON/ARRAYS/A(130)
      COMMON/NODEA/NA(420)
      COMMON/NODEB/NB(420)
      COMMON/COND/G(420)
      COMMON/MODEL/ANAME(1)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
```

```
      COMMON/BNDRY/NDATA1,NDATA2,TIMHR1(50),TIMHR2(50),SOLAR(50),
     &             TMPAMB(50),WNDS(50),WNDD(50)
      COMMON/COEFF/SOLARB(50),SOLARC(50),SOLARD(50),
     &             TAMBB(50),TAMBC(50),TAMBD(50),
     &             WNDSB(50),WNDSC(50),WNDSD(50),
     &             WNDDB(50),WNDDC(50),WNDDD(50)
      COMMON/TSTVAL/ITRSUM,FLRATE,IFRST,FLORAT,DELTAT,CTRLLR
      DATA DTIME0,TFLG/0.0,.TRUE./
C
      CALL GTTIME(I1,IDAY,IMON,IYR,IHR,IMIN,ISEC)
C
      ATIME=IHR+(IMIN/100.0)+(ISEC/10000.0)
      CALL BA6DEC(ATIME,IHR0,IMIN0,ISEC0,DTIME)
C
      IF(TFLG)DTIME0=DTIME
      TFLG=.FALSE.
      IF(DTIME.GE.DTIME0)GO TO 10
C
      BTIME=24.0-DTIME0+DTIME
C
      GO TO 20
10    BTIME=DTIME-DTIME0
C
20    CALL DECBA6(BTIME,IHR1,IMIN1,ISEC1,TIME6)
      DTIME0=DTIME
C
      IF(ITRSUM.GT.10000)ITRSUM=ITRSUM-10000
C
      WRITE(1,1000)IDAY,IMON,IYR,IHR,IMIN,ISEC,IMIN1,ISEC1,ITRSUM,
     & TIMEN,DELTAT,TNEW(52),TNEW(120),FLORAT,DTIMEI,DTIMEU
1000  FORMAT(1H ,I2,'/',I2,'/',I2,2X,I2,':',I2,'.',I2,2X,I2,'.',
     & I2,2X,I5,2X,F4.1,4(2X,F6.1),2X,2G14.7)
C
      WRITE(7)TIMEN,FLORAT,TNEW
C
      RETURN
```

```
      END
C
      BLOCK DATA BLKDAT
      LOGICAL CTRLLR
      COMMON/TEMPO/TOLD(130)
      COMMON/TEMPN/TNEW(130)
      COMMON/CAP/C(130)
      COMMON/HEAT/Q(130)
      COMMON/ARRAYS/A(130)
      COMMON/NODEA/NA(420)
      COMMON/NODEB/NB(420)
      COMMON/COND/G(420)
      COMMON/MODEL/ANAME(9)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
      COMMON/BNDRY/NDATA1,NDATA2,TIMHR1(50),TIMHR2(50),SOLAR(50),
     &             TMPAMB(50),WNDS(50),WNDD(50)
      COMMON/COEFF/SOLARB(50),SOLARC(50),SOLARD(50),
     &             TAMBB(50),TAMBC(50),TAMBD(50),
     &             WNDSB(50),WNDSC(50),WNDSD(50),
     &             WNDDB(50),WNDDC(50),WNDDD(50)
      COMMON/TSTVAL/ITRSUM,FLRATE,IFRST,FLORAT,DELTAT,CTRLLR
      DATA    ANAME/'PPG ','ALUM','INUM',' COL','LECT','OR: ',
     &             ' TES','T CA','SE  '/
      DATA    NNOD    ,NCOND  ,NAMSIZ ,NLOOP  ,LOOPCT ,OUTPUT ,TIMEM
     &       /0       ,0      ,9      ,50     ,0       ,0.1     ,0.0/
      DATA    TIMEND  ,TIMEO  ,DTIMEI ,DTIMEU ,DRLXCA ,DRLXCC ,NDRLXC
     &       /19.0    ,5.0    ,3.333333E-2    ,0.0     ,0.05    ,0.0,0/
      DATA    TIMEN   /5.0/
      DATA ITRSUM/0/
      END
```

FILE: THRMLLIB.REL

```
      SUBROUTINE HPRINT(IFILE)
C
C PURPOSE:       TO PROVIDE A HEADER PRINTOUT OF ALL
C                VARIABLES  USED IN THE THERMAL ANALYZER PROGRAM.
C
C INPUTS:        IFILE-THE UNIT NUMBER TO WRITE TO.
C
C OUTPUTS:       FORMATTED OUTPUT OF CERTAIN CONSTANTS WITH ALPHA
C                DESCRIPTORS.
C
C SUBROUTINES REFERENCED:        SKIP.
C
C RESTRICTIONS: REQUIRES NAMSIZ TO BE SET TO THE NUMBER OF 4 ELEMENT
C                CHARACTERS IN ANAME( ).
C
C START
      COMMON/MODEL/ANAME(1)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
C
C OUTPUT STANDARD HEADING
C
      CALL SKIP(IFILE,2)
      WRITE(IFILE,1000)(ANAME(I),I=1,NAMSIZ)
1000  FORMAT(5X,30A4//)
      WRITE(IFILE,1010)TIMEN,LOOPCT,DRLXCC,DTIMEU,OUTPUT
1010  FORMAT(1H ,15X,'TIMEN =',G10.4,'   LOOPCT=',5X,I5,'   DRLXCC=',
     &       G10.4,'   DTIMEU=',G10.4,'   OUTPUT=',G10.4)
      WRITE(IFILE,1020)TIMEO,NLOOP,NDRLXC,DTIMEI,TIMEND
1020  FORMAT(1H ,15X,'TIMEO =',G10.4,'   NLOOP =',5X,I5,'   NDRLXC=',
     &       5X,I5,'   DTIMEI=',G10.4,'   TIMEND=',G10.4)
      WRITE(IFILE,1030)TIMEM,NAMSIZ,DRLXCA,NNOD,NCOND
1030  FORMAT(1H ,15X,'TIMEM =',G10.4,'   NAMSIZ=',5X,I5,'   DRLXCA=',
     &       G10.4,'   NNOD  =',5X,I5,'   NCOND =',5X,I5)
      RETURN
```

```
      END
C
      SUBROUTINE TPRINT(IFILE)
C
C PURPOSE:       TO PROVIDE A PRINTOUT OF THE TNEW( ) ARRAY FROM THE
C                THERMAL ANALYZER PROGRAM.
C
C INPUTS:        IFILE-THE UNIT NUMBER TO WRITE TO.
C
C OUTPUTS:       FORMATTED  OUTPUT OF TNEW( ) TO IFILE.
C
C SUBROUTINES REFERENCED:        SKIP.
C
C RESTRICTIONS: REQUIRES NNOD TO BE SET TO THE NUMBER OF NODES IN THE
C                PROBLEM.
C
C START
C
      COMMON/TEMPN/TNEW(1)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
C
      CALL SKIP(IFILE,1)
      WRITE(IFILE,1120)
1120  FORMAT(53X,'NODE TEMPERATURES(NEW)')
      DO 10 J=1,NNOD,6
      LIM=J+5
      IF(LIM.GT.NNOD)LIM=NNOD
10    WRITE(IFILE,1130)(I,TNEW(I),I=J,LIM)
1130  FORMAT(7X,6('T(',I3,')=',G10.3,3X))
      RETURN
      END
C
      SUBROUTINE QPRINT(IFILE)
C
```

```
C PURPOSE:       TO PROVIDE PRINTOUTS OF THE NODE HEAT SOURCES/SINKS
C                FROM THE THERMAL ANALYZER PROGRAM.
C
C INPUTS:        IFILE-THE UNIT NUMBER TO WRITE TO.
C
C OUTPUTS:       FORMATTED OUTPUT OF Q( ) TO IFILE.
C
C SUBROUTINES REFERENCED:        SKIP.
C
C RESTRICTIONS: REQUIRES NNOD TO BE SET TO THE NUMBER OF NODES IN THE
C                PROBLEM.
C
C START
C
        COMMON/HEAT/Q(1)
        COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &               TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &               ,TIMEN
C
        CALL SKIP(IFILE,1)
        WRITE(IFILE,1180)
1180    FORMAT(54X,'NODE HEAT SOURCE/SINK')
        DO 40 J=1,NNOD,6
        LIM=J+5
        IF(LIM.GT.NNOD)LIM=NNOD
40      WRITE(IFILE,1190)(I,Q(I),I=J,LIM)
1190    FORMAT(7X,6('Q(',I3,')=',G10.4,3X))
C
        RETURN
        END
C
        SUBROUTINE CPRINT(IFILE)
C
C PURPOSE:       TO PROVIDE PRINTOUTS OF THE NODE CAPACITANCES
C                FROM THE THERMAL ANALYZER PROGRAM.
C
```

```
C INPUTS:        IFILE-THE UNIT NUMBER TO WRITE TO.
C
C OUTPUTS:       FORMATTED OUTPUT OF C( ) TO IFILE.
C
C SUBROUTINES REFERENCED:        SKIP.
C
C RESTRICTIONS: REQUIRES NNOD TO BE SET TO THE NUMBER OF NODES IN THE
C                PROBLEM.
C
C START
C
        COMMON/CAP/C(1)
        COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &               TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &               ,TIMEN
C
        CALL SKIP(IFILE,1)
        WRITE(IFILE,1160)
1160    FORMAT(56X,'NODE CAPACITANCES')
        DO 30 J=1,NNOD,6
        LIM=J+5
        IF(LIM.GT.NNOD)LIM=NNOD
30      WRITE(IFILE,1170)(I,C(I),I=J,LIM)
1170    FORMAT(7X,6('C(',I3,')=',G12.4,1X))
C
        RETURN
        END
C
        SUBROUTINE GPRINT(IFILE)
C
C PURPOSE:       TO PROVIDE PRINTOUTS OF THE NODE TO NODE
C                CONDUCTOR VALUES FROM THE THERMAL ANLYZER PROGRAM.
C
C INPUTS:        IFILE-THE UNIT NUMBER TO WRITE TO.
C
C OUTPUTS:       FORMATTED OUTPUT OF NA( ), NB( ),
```

```
C                AND G( ) TO IFILE.
C
C SUBROUTINES REFERENCED:        SKIP
C
C RESTRICTIONS: REQUIRES NCOND TO BE SET TO THE NUMBER OF CONDUCTOR
C                VALUES IN THE PROBLEM.
C
C START
C
C
        COMMON/NODEA/NA(1)
        COMMON/NODEB/NB(1)
        COMMON/COND/G(1)
        COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &                TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &                ,TIMEN
C
        CALL SKIP(IFILE,1)
        WRITE(IFILE,1200)
1200    FORMAT(52X,'NODE TO NODE CONDUCTORS')
        WRITE(IFILE,1210)
1210    FORMAT(6X,'G#',3X,'NA',5X,'NB',5X,'VALUE',5X,'G#',3X,'NA',5X,
     &  'NB',5X,'VALUE',5X,'G#',3X,'NA',5X,'NB',5X,'VALUE',5X,'G#',3X,
     &  'NA',5X,'NB',5X,'VALUE')
        DO 50 J=1,NCOND,4
        LIM=J+3
        IF(LIM.GT.NCOND)LIM=NCOND
50      WRITE(IFILE,1220)(I,NA(I),NB(I),G(I),I=J,LIM)
1220    FORMAT(4X,4(I4,1X,I4,3X,I4,1X,G10.4,2X))
        RETURN
        END
C
        SUBROUTINE INTRPL(W,X,Y,Z)
C
C PURPOSE:       TO PROVIDE A LINEAR INTERPOLATION OF X,THE INDEPENDENT
C                ARRAY, AND Y , THE DEPENDENT ARRAY USING W TO INTER-
```

```
C                POLATE AND Z AS THE VARIABLE TO RECIEVE THE INERPOLATED
C                RESULT.
C
C INPUTS:        W-INDEPENDENT VALUE FOR WHICH AN INTERPOLATED RESULT
C                  IS NEEDED.
C                X-ARRAY CONTAINING THE INDEPENDENT VALUES.
C                Y-ARRAY CONTAINING THE DEPENDENT VALUES.
C
C OUTPUTS:       Z-VARIABLE TO RECIEVE THE INTERPOLATED RESULT.
C
C SUBROUTINES REFERENCED:        NONE
C
C RESTRICTIONS: X(1) AND Y(1) MUST CONTAIN THE NUMBER OF ELEMENTS
C                IN THE ARRAYS.  X AND Y DATA VALUES MUST CORRESPOND.
C
C START
C
      DIMENSION X(1),Y(1)
      LAST=IFIX(X(1))+1
      IFIRST=2
      IF(W.LT.(X(IFIRST)))GO TO 100
      IF(W.GT.(X(LAST)))GO TO 200
      DO 10 I=2,LAST
      IF((W.GT.X(I)).AND.(W.LT.X(I+2)))GO TO 300
10    CONTINUE
100   Z=Y(IFIRST)
      GO TO 9999
200   Z=Y(LAST)
      GO TO 9999
300   Z=((Y(I+2)-Y(I))/(X(I+2)-X(I)))*(W-X(I))+Y(I)
9999  RETURN
      END
C
      SUBROUTINE GETINP(IFILE1,IFILE2)
C
C PURPOSE:       THIS ROUTINE READS INPUT DATA FROM IFILE1 FOR THE
```

```
C                THERMAL ANALYZER PROGRAM.  INPUT DATA CONSISTS OF A
C                NODE DATA BLOCK AND A CONDUCTOR DATA BLOCK.  THE NODE
C                BLOCK CONTAINS THE NODE NUMBER,INITIAL TEMPERATURE,
C                CAPACITANCE, AND HEAT SOURCE/SINK.  THE CONDUCTOR
C                DATA BLOCK CONTAINS THE ADJOINING NODE NUMBERS AND
C                THE CONDUCTOR VALUE.
C                DATA IS ALSO READ IN FROM IFILE2.  THIS DATA COMPRISES
C                THE BOUNDARY CONDITIONS THAT ARE TO BE LOADED INTO
C                ARRAYS TIMHR,SOLAR,TMPAMB,WNDS,AND WNDD.  NOTE THAT
C                ONLY 50 ELEMENTS CAN BE LOADED INTO THESE ARRAYS.
C
C INPUTS:        IFILE1-THE FILE TO READ THE COLLECTOR DATA FROM.
C                IFILE2-THE FILE TO READ BOUNDARY CONDITIONS FROM.
C
C OUTPUTS:       FILLS THE TNEW( ),C( ),Q( ),NA( ),NB( ) AND G( )
C                ARRAYS.  ALSO SETS NNOD AND NCOND TO THE NUMBER OF
C                NODES AND NUMBER OF CONDUCTORS RESPECTIVELY.  THE LAST
C                DATA ITEMS OF TNEW( ),C( ), Q( ), AND G( )
C                CONTAIN 32767.0.  THE LAST DATA ITEMS OF NA( ) AND NB( )
C                CONTAIN 32767.
C                FILLS THE TIMHR( ),SOLAR( ),TMPAMB( ),WNDS( ),WNDD( )
C                ARRAYS.  ALSO SETS NDATA TO BE THE NUMBER OF DATA
C                VALUES READ IN.
C
C SUBROUTINES REFERENCED:        UNPACK
C
C RESTRICTIONS: INPUT DATA MAY CONTAIN COMMENT STATEMENTS.  THE LAST
C                DATA ITEM OF EACH BLOCK MUST CONTAIN AN E IN COLUMN 1
C                (I.E. END).  ONLINE COMMENTS MAY ALSO BE INCLUDED AFTER
C                ALL DATA ITEMS ARE FILLED ON A SINGLE LINE.
C                INPUT DATA MUST START WITHIN THE FIRST 5 COLUMNS.
C                MAXIMUM NUMBER OF DATA ITEMS THAT CAN BE READ INTO
C                THE BOUNDARY ARRAYS IS 50.
C
C START
C
```

```
      LOGICAL CMMNT,END,COL1,FIRST,IBUF2(128)
      COMMON/TEMPN/TNEW(1)
      COMMON/CAP/C(1)
      COMMON/HEAT/Q(1)
      COMMON/ARRAYS/A(1)
      COMMON/NODEA/NA(1)
      COMMON/NODEB/NB(1)
      COMMON/COND/G(1)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
      COMMON/BNDRY/NDATA1,NDATA2,TIMHR1(50),TIMHR2(50),SOLAR(50),
     &             TMPAMB(50),WNDS(50),WNDD(50)
C
      CMMNT=Z'43'
      END=Z'45'
      NNOD=0
      NCOND=0
C
C SET FIRST PASS FLAG FOR SUBROUTINE UNPACK
C
      FIRST=.TRUE.
C
C PROCESS THE DATA BLOCK
C
100   CALL UNPACK(IFILE1,IBUF2,FIRST)
C
C CHECK FOR A COMMENT OR END MARKER
C
      DECODE(IBUF2,200)COL1
200   FORMAT(A1)
      IF(COL1.EQ.CMMNT)GO TO 100
      IF(COL1.EQ.END)GO TO 500
C
C OTHERWISE READ DATA AND THEN PROCESS NEXT RECORD
C
```

```
      DECODE(IBUF2,300)I,TNEW(I),C(I),Q(I),A(I)
300   FORMAT(I5,4G14.7)
      NNOD=NNOD+1
      GO TO 100
C
C END OF NODE DATA BLOCK; SET LAST ELEMENTS TO 32767.
C
500   I=NNOD+1
      TNEW(I)=32767.
      C(I)=32767.
      Q(I)=32767.
      A(I)=32767.
C
C PROCESS THE CONDUCTOR DATA BLOCK
C
525   NCOND=NCOND+1
550   CALL UNPACK(IFILE1,IBUF2,FIRST)
C
C CHECK FOR A COMMENT OR END MARKER
C
      DECODE(IBUF2,600)COL1
600   FORMAT(A1)
      IF(COL1.EQ.CMMNT)GO TO 550
      IF(COL1.EQ.END)GO TO 900
C
C OTHERWISE READ DATA AND PROCESS NEXT RECORD
C
      DECODE(IBUF2,700)NA(NCOND),NB(NCOND),G(NCOND)
700   FORMAT(2I5,G10.4)
      GO TO 525
C
C END OF CONDUCTOR DATA BLOCK; SET LAST ELEMENTS TO 32767
C
900   NA(NCOND)=32767
      NB(NCOND)=32767
      G(NCOND)=32767.
```

```
          NCOND=NCOND-1
C
C
C LOAD THE BOUNDARY DATA ARRAYS
C
          FIRST=.TRUE.
          NDATA1=0
C
C READ IN RECORD AND CHECK FOR A COMMENT OR AN END STATEMENT.
C
1000      CALL UNPACK(IFILE2,IBUF2,FIRST)
          DECODE(IBUF2,200)COL1
          IF(COL1.EQ.CMMNT)GO TO 1000
          IF(COL1.EQ.END)GO TO 9100
C
C OTHERWISE READ DATA IN
C
          NDATA1=NDATA1+1
          DECODE(IBUF2,1010)TIMHR1(NDATA1),SOLAR(NDATA1)
1010      FORMAT(2G14.7)
          GO TO 1000
C
9100      NDATA2=0
C
C READ IN RECORD AND CHECK FOR A COMMENT OR AN END STATEMENT.
C
1500      CALL UNPACK(IFILE2,IBUF2,FIRST)
          DECODE(IBUF2,200)COL1
          IF(COL1.EQ.CMMNT)GO TO 1500
          IF(COL1.EQ.END)GO TO 9999
C
C OTHERWISE READ DATA IN
C
          NDATA2=NDATA2+1
          DECODE(IBUF2,1020)TIMHR2(NDATA2),TMPAMB(NDATA2),
     &                      WNDS(NDATA2),WNDD(NDATA2)
```

```
1020    FORMAT(3G14.7,A4)
        GO TO 1500
C
9999    RETURN
        END
C
        SUBROUTINE APRINT(IFILE,IFIRST,LAST)
C
C PURPOSE:      TO PROVIDE PRINTOUTS OF THE A ARRAY
C               FROM THE THERMAL ANALYZER PROGRAM.
C
C INPUTS:       IFILE-THE UNIT NUMBER TO WRITE TO.
C               IFIRST-FIRST ELEMENT TO OUTPUT
C               LAST-LAST ELEMENT TO OUTPUT
C
C OUTPUTS:      FORMATTED OUTPUT OF A( ) TO IFILE.
C
C SUBROUTINES REFERENCED:       SKIP.
C
C RESTRICTIONS: REQUIRES NNOD TO BE SET TO THE NUMBER OF NODES IN THE
C               PROBLEM.
C
C START
C
        COMMON/ARRAYS/A(1)
        COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &              TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &              ,TIMEN
C
        CALL SKIP(IFILE,1)
        WRITE(IFILE,1180)
1180    FORMAT(60X,'ARRAY A( )')
        DO 40 J=IFIRST,LAST,6
        LIM=J+5
        IF(LIM.GT.LAST)LIM=LAST
40      WRITE(IFILE,1190)(I,A(I),I=J,LIM)
```

```
1190    FORMAT(7X,6('A(',I3,')=',G10.4,3X))
C
        RETURN
        END
C
        SUBROUTINE SOLVE
C
C PURPOSE:       THIS SUBROUTINE ITERATES A BLOCK OF DATA USING THE
C                CRANK-NICHOLSON METHOD UNTIL A MAXIMUM TEMPERATURE
C                DIFFERENCE, BETWEEN ITERATIONS IS MET OR A LOOP COUNT
C                IS EXCEEDED (I.E. LOOPCT.GT.NLOOP) ONLY DIFFUSION
C                NODES ARE ITERATED.  PRNOUT IS CALLED UPON ENTRY
C                TO PROVIDE CONFIRMATION OF INITIAL VALUES.  CONTROL
C                PARAMETERS AND CONSTANTS ARE CALCULATED THEN PRE
C                IS CALLED TO SET HEAT FLUXES AND OTHER TIME,TEMPERATURE
C                CAPACITANCE ETC., RELATED VALUES.  THE BLOCK IS THEN
C                ITERATED USING A SUCCESSIVE POINT ITERATION SCHEME UNTIL
C                THE DRLXCA OR LOOPCT PARAMETERS ARE MET.  POST IS THEN
C                CALLED TO ADJUST VALUES IF NECESSARY.  PRNOUT IS CALLED
C                IF THE OUTPUT INTERVAL IS MET.  THIS PROCESS IS
C                CONTINUED UNTIL TIMEND IS MET.
C
C INPUTS:        VARIABLES IN THE COMMON BLOCKS ARE USED FOR INPUTS.
C
C OUTPUTS:       ALL OUTPUT VALUES (THE TEMPERATURE VALUES) ARE IN THE
C                COMMON BLOCKS.
C
C RESTRICTIONS: CONTROL CONSTANTS TIMEN,TIMEO,TIMEND,DRLXCA,DTIMEI
C                NLOOP,NNOD,AND NCOND MUST BE SET .
C
C START
        COMMON/TEMPO/TOLD(1)
        COMMON/TEMPN/TNEW(1)
        COMMON/CAP/C(1)
        COMMON/HEAT/Q(1)
        COMMON/ARRAYS/A(1)
```

```
      COMMON/NODEA/NA(1)
      COMMON/NODEB/NB(1)
      COMMON/COND/G(1)
      COMMON/MODEL/ANAME(1)
      COMMON/CONST/NNOD,NCOND,NAMSIZ,NLOOP,LOOPCT,OUTPUT,TIMEM,
     &             TIMEND,TIMEO,DTIMEI,DTIMEU,DRLXCA,DRLXCC,NDRLXC
     &             ,TIMEN
C
C ***************************** FIRST ENTRY TO THIS ROUTINE ********************
C
C INITIALIZE VARIABLES, CALL PRE TO SET BOUNDARY CONDITIONS AND CALL
C PRNOUT TO PRINT THE INITIAL CONDITIONS
C
      TIMEN=TIMEO
      CALL PRE
      CALL PRNOUT
      OUTNXT=TIMEO+OUTPUT
C
C ******* THIS IS THE ENTRY POINT FOR EACH NEW TIME STEP **********************
C
C TRANSFER THE TEMPERATURES FROM THE CURRENT TIME STEP INTO THE ARRAY
C FOR THE OLD TIME STEP I.E. THE NEW TEMPERATURES BECOME THE OLD TEMPERATURES
C FOR THE NEXT TIME STEP.  ALSO CALCULATE THE TIME INTERVAL AND WHEN PRNOUT
C IS TO BE CALLED AND SET UP VIA PRE ALL THE BOUNDARY CONDITIONS AND
C OTHER VARIABLES FOR THIS TIME STEP.
C
100   DO 110 I=1,NNOD
110   TOLD(I)=TNEW(I)
C
      DTIMEU=DTIMEI
      IF(((TIMEO+DTIMEU).LT.OUTNXT).AND.((TIMEO+2.0*DTIMEU).GT.
     & OUTNXT))DTIMEU=(OUTNXT-TIMEO)/2.
      IF((TIMEO+DTIMEU).GT.OUTNXT)DTIMEU=OUTNXT-TIMEO
      IF(DTIMEU.LE.1.0E-04)GO TO 9000
      TIMEN=TIMEO+DTIMEU
      DTIMD2=DTIMEU/2.0
```

```
          TIMEM=TIMEN+DTIMD2
          IFLAG=1
C
C
C ********************************* SOLVE THE NETWORK OF NODES ******************
C
C SOLVE THE SYSTEM OF EQUATIONS BY RELAXATION METHODS
C
          DO 1000 LOOPCT=1,NLOOP
C
C FIRST OR SECOND PASS THROUGH NETWORK?
C
          GO TO (1200,2000),IFLAG
1200      IFLAG=2
          CALL PRE
          DRLXCC=0.0
C
C CONVERT TEMPERATURES TO RANKINE
C
          DO 1250 I=1,NNOD
          TOLD(I)=TOLD(I)+460.0
1250      TNEW(I)=TNEW(I)+460.0
C
C FIRST PASS: PERFORM FIRST ITERATION, ALSO SET UP CONSTANTS FOR
C NEXT ITERATIONS
C
          DO 1500 I=1,NNOD
C
C DON'T ITERATE BOUNDARY NODES; BOUNDARY NODES HAVE NEGATIVE CAPACITANCE
C
          IF(C(I).LT.0.0)GO TO 1500
C
C SET UP TO ITERATE NODE
C
          C(I)=C(I)/DTIMD2
          CAP=C(I)
```

```
      QOI=2.0*Q(I)+TOLDI*CAP
      TOLDI=TOLD(I)
      HTSUM=0.0
      SLGIJ=0.0
      SRGIJ=0.0
      IROW=0
C
C FIND AN ADJOINING NODE, RETRIEVE IT'S CONDUCTOR AND SUM INTO QUARTIC
C CONSTANTS.
C
1550  CALL SEARCH(I,IROW,NODE)
      IF(IROW.EQ.0)GO TO 1560
      TJO=TOLD(NODE)
      TJN=TNEW(NODE)
      GIJ=G(IROW)
C
C IF CONDUCTOR IS NEGATIVE THEN IT IS A RADIATION CONDUCTOR
C THAT IS, IT'S NOT A LINEAR CONDUCTOR
C
      IF(GIJ.LT.0.0)GO TO 1555
C
C OTHERWISE SUM AS A LINEAR CONDUCTOR
C
      SLGIJ=SLGIJ+GIJ
      QOI=QOI+GIJ*TJN
      HTSUM=HTSUM+GIJ*TJO
C
C GO TO SEARCH FOR NEXT ADJOINING NODE
C
      GO TO 1550
C
C SUM AS A RADIATION CONDUCTOR
C
1555  SRGIJ=SRGIJ-GIJ
      TJN2=TJN*TJN
      QOI=QOI-GIJ*TJN2*TJN2
```

```
          TJO2=TJO*TJO
          HTSUM=HTSUM-GIJ*TJO2*TJO2
C
C GO TO SEARCH FOR NEXT ADJOINING NODE
C
          GO TO 1550
C
C AT THIS POINT THERE ARE NO MORE ADJOINING NODES; SET UP CONSTANTS
C THEN SOLVE THE QUARTIC OR LINEAR EQUATION
C
1560      IF(SRGIJ.EQ.0.0)GO TO 1570
C
          Q(I)=TOLDI*(TOLDI*TOLDI*TOLDI*SRGIJ+SLGIJ)-QOI
          C1=(CAP+SLGIJ)/SRGIJ
          Q2=-(C1*C1)*0.0625
          P3=(Q(I)-HTSUM)/SRGIJ*0.3333333
C
C DIVIDE BY 1E+40 TO KEEP EXPONENT WITHIN RANGE
C
          P3=(Q2*1E-20)*(Q2*1E-20)-(P3*1E-20)*(P3*1E-20)*P3
          IF(P3.GT.0.0)GO TO 1575
          TNEWI=0.0
          GO TO 1580
C
C RESTORE P3 TO IT'S FULL VALUE
C
1575      P3=ESQRT(P3)*1E+20
          A1=ESQRT(CBRT(P3-Q2)-CBRT(P3+Q2))
          A2=-(A1*A1)+C1*0.7071068/A1
          TNEWI=0.7071068*(-A1+ESQRT(A2))
          GO TO 1580
1570      IF(SLGIJ.EQ.0.0)GO TO 1500
C
C THIS EQUATION IS USED IF THERE ARE NO RADIATION CONDUCTORS I.E. SOLVE
C BY A SIMPLE LINEAR EQUATION.
C
```

```
        Q(I)=-TOLDI*SLGIJ+QOI
        TNEWI=(Q(I)+HTSUM)/(CAP+SLGIJ)
C
C CALCULATE THE TEMPERATURE CHANGE,SET THE NEW TEMPERATURE, AND CHECK
C IF THIS NODE HAS THE LARGEST CHANGE BETWEEN ITERATIONS
C
C
1580    DELTAT=TNEWI-TOLDI
        TNEW(I)=TNEWI
        IF(ABS(DRLXCC).GE.ABS(DELTAT))GO TO 1500
        DRLXCC=DELTAT
        NDRLXC=I
1500    CONTINUE
C
C THIS LOOP IS USED FOR THE REMAINDER OF THE ITERATIONS FOR A TIME STEP
C THIS LOOP SAVES TIME BECAUSE SOME CONSTANTS WHICH DO NOT CHANGE
C DURING A TIME STEP, WERE CALCULATED PREVIOUSLY.
C
2000    DO 3000 I=1,NNOD
C
C DON'T ITERATE BOUNDARY NODES, BOUNDARY NODES HAVE NEGATIVE CAPACITANCE.
C
        IF(C(I).LT.0.0)GO TO 3000
C
C SET UP TO ITERATE NODE
C
        CAP=C(I)
        TOLDI=TNEW(I)
        HTSUM=0.0
        SLGIJ=0.0
        SRGIJ=0.0
        IROW=0
C
C FIND AN ADJOINING NODE, RETRIEVE IT'S CONDUCTOR AND SUM INTO QUARTIC
C CONSTANTS.
C
```

```
3100     CALL SEARCH(I,IROW,NODE)
         IF(IROW.EQ.0)GO TO 3300
         TJN=TNEW(NODE)
         GIJ=G(IROW)
C
C IF CONDUCTOR IS NEGATIVE THEN IT IS A RADIATION CONDUCTOR- NOT A
C LINEAR CONDUCTOR
C
         IF(GIJ.LT.0.0)GO TO 3200
C
C OTHERWISE SUM AS A LINEAR CONDUCTOR
C
         SLGIJ=SLGIJ+GIJ
         HTSUM=HTSUM+GIJ*TJN
C
C GO TO SEARCH FOR NEXT ADJOINING NODE
C
         GO TO 3100
C
C AT THIS POINT THERE ARE NO MORE ADJOINING NODES
C
3200     SRGIJ=SRGIJ-GIJ
         TJN2=TJN*TJN
         HTSUM=HTSUM-GIJ*TJN2*TJN2
         GO TO 3100
C
C SOLVE EITHER THE QUARTIC OR THE LINEAR EQUATION
C
3300     IF(SRGIJ.EQ.0.0)GO TO 3310
C
         C1=(CAP+SLGIJ)/SRGIJ
         Q2=-(C1*C1)*0.0625
         P3=(Q(I)-HTSUM)/SRGIJ*0.3333333
C
C DIVIDE BY 1E+40 TO KEEP EXPONENT WITHIN RANGE
```

```
C
        P3=(Q2*1E-20)*(Q2*1E-20)-(P3*1E-20)*(P3*1E-20)*P3
        IF(P3.GT.0.0) GO TO 3320
        TNEWI=0.0
        GO TO 3330
C
C RESTORE P3 TO ITS FULL VALUE
C
3320    P3=ESQRT(P3)*1E+20
        A1=ESQRT(CBRT(P3-Q2)-CBRT(P3+Q2))
        A2=-(A1*A1)+C1*0.7071068/A1
        TNEWI=0.7071068*(-A1+ESQRT(A2))
        GO TO 3330
C
C THIS EQUATION IS USED IF THERE ARE NO RADIATION CONDUCTORS I.E. SOLVE
C BY A SIMPLE LINEAR EQUATION.
C
3310    IF(SLGIJ.EQ.0.0)GO TO 3000
        TNEWI=(Q(I)+HTSUM)/(CAP+SLGIJ)
C
C COMPUTE TEMPERATURE CHANGE AND SET NEW TEMPERATURE
C
3330    DELTAT=TNEWI-TOLDI
        TNEW(I)=TNEWI
        IF(ABS(DRLXCC).GE.ABS(DELTAT))GO TO 3000
        DRLXCC=DELTAT
        NDRLXC=I
3000    CONTINUE
        IF(ABS(DRLXCC).LE.DRLXCA)GO TO 4000
1000    CONTINUE
C
C BLOCK ITERATION IS COMPLETE,RESTORE C(I) AND TNEW(I)
C
4000    DO 4100 I=1,NNOD
        IF(C(I).LT.0.0)GO TO 4100
        C(I)=C(I)*DTIMD2
```

```
4100     CONTINUE
         DO 5000 I=1,NNOD
         TOLD(I)=TOLD(I)-460.0
5000     TNEW(I)=TNEW(I)-460.0
         CALL POST
C
C CHECK IF TIME TO PRINT
C
         TIMEO=TIMEN
         TLRNCE=DTIMEU/3.
         IF(.NOT.((OUTNXT.LT.TIMEN+TLRNCE).AND.(OUTNXT.GT.TIMEN-TLRNCE))
     &   GO TO 9999
9000     CALL PRNOUT
         OUTNXT=TIMEO+OUTPUT
C
C CHECK FOR TIME TO STOP
C
9999     IF(TIMEN+TLRNCE.LE.TIMEND)GO TO 100
         RETURN
         END
C
C
         SUBROUTINE STPLHC(WNDSPD)
C
C PURPOSE:       THIS SUBROUTINE CALCULATES THE CONDUCTANCE TERMS
C                FOR CONVECTION BETWEEN FLAT PLATES.  EQUATIONS
C                4.11.1, 4.11.2, 4.11.9, AND 4.14,4 OF DUFFIE AND
C                BECKMAN ARE USED.  A PLATE ANGLE OF 70 DEGREES
C                IS USED.
C
C INPUTS:        WNDSPD - WIND SPEED IN (MILES PER HOUR); USED IN
C                CALCULATING THE CONVECTION FROM THE TOP COVER
C                PLATE TO THE AMBIENT
C                ALL OTHER INPUTS ARE FROM THE COMMON BLOCKS NODEA,
C                NODEB, TEMPN, ARRAYS, AND COND.
C
```

```
C OUTPUTS:       ALL OUTPUTS ARE TO THE COMMON BLOCK COND.
C
C SUBROUTINES REFERENCED:        NONE
C
C RESTRICTIONS: ASSUMES CONSTANT PROPERTIES FOR AIR BETWEEN THE PLATES
C               ALSO, BECAUSE PLATE SPACING IS SMALL (0.375 INCHES
C               MINIMUM TO 0.5 INCHES MAXIMUM) CONVECTION IS ASSUMED
C               FOR THE SMALLEST AREA, I.E. CONVECTION IS CONTAINED
C               WITHIN A COLUMN EXTENDING PERPENDICULAR FROM THE
C               LOWER NODE (SMALLEST AREA) TO THE HIGHER PLATE.
C               THE SMALLEST NODE AREA IS USED IN CALCULATING THE
C               CONDUCTANCE TERM.
C
C START
C
      COMMON/TEMPN/TNEW(1)
      COMMON/ARRAYS/A(1)
      COMMON/NODEA/NA(1)
      COMMON/NODEB/NB(1)
      COMMON/COND/G(1)
      DATA GRAVIT/32.1739/,AKAIR/0.0154/,BETA/1.79E-03/,
     1     DVISC/0.180E-03/,FACTOR/4.6777E-02/
C
C SET CONDUCTANCES FOR COLLECTOR PLATE TO FIRST GLASS COVER PLATE
C
      IFIRST=272
      LAST=327
      PLSPAC=0.5/12.0
C
10    DO 100 I=IFIRST,LAST
      INA=NA(I)
      INB=NB(I)
      DELTA=TNEW(INA)-TNEW(INB)
C
      IF(DELTA.GT.0.0)GO TO 50
C
```

```
C CONVECTION IS SUPPRESSED;  SET CONDUCTANCE BASED ON CONDUCTION ONLY
C
        G(I)=A(INA)*AKAIR/PLSPAC
        GO TO 100
C
C OTHERWISE SET CONDUCTANCE VALUE BASED ON FREE CONVECTION
C
50      GRASHF=(GRAVIT*BETA*DELTA*PLSPAC*PLSPAC*PLSPAC)/
     1          (DVISC*DVISC)
C
C CALCULATE THE NUSSELT NUMBER
C
C       ANUSLT=FACTOR*(GRASHF**0.33333)
        ANUSLT=FACTOR*CBRT(GRASHF)
C
C SET THE CONDUCTANCE VALUE BASED ON CONVECTION AND CONDUCTION
C
        G(I)=(ANUSLT+1)*A(INA)*AKAIR/PLSPAC
100     CONTINUE
C
        IF(LAST.EQ.339)GO TO 600
C
C SET CONDUCTANCES BETWEEN COVER PLATES
C
        IFIRST=328
        LAST=339
        PLSPAC=0.375/12.0
        GO TO 10
600     CONTINUE
C
C CALCULATE LOSSES FROM TOP PLATE VIA WIND USING EQUATION 4.14.4
C
        HWIND=1.00+1.498*WNDSPD
        H=HWIND*A(117)
        G(340)=H
        G(341)=H
```

```
      G(342)=H
C
      RETURN
      END
C
      SUBROUTINE STFLOW(FLORAT)
C
C PURPOSE:       THIS SUBROUTINE SETS TUBE NODE CONDUCTANCES THAT
C                ARE A FUNCTION OF FLOW RATE.  EQUATION USED IS:
C                G=MASS FLOW RATE * HEAT CAPACITY.
C
C INPUTS:        FLOW-R-FLOW RATE (LBM/HOUR) OF BOTH HALVES OF COLLECTOR
C
C OUTPUTS:       ALL OUTPUTS ARE TO THE COND COMMON BLOCK.
C
C SUBROUTINES REFERENCED:        NONE.
C
C RESTRICTIONS: CONDUCTOR NUMBERS ARE REFERENCED BY THEIR POSITION
C                RELATIVE TO THE START OF THE COND COMMON BLOCK.  ANY
C                CHANGES TO THE CONDUCTOR BLOCK MAY REQUIRE THIS
C                SUBROUTINE TO BE UPDATED.
C
C START
C
      COMMON/COND/G(1)
C
C FLOW IS  ACTUAL FLOW RATE (LBS/HOUR) 13=NUMBER OF TUBES
C
      FLOW=FLORAT/13.0
      IF(FLOW.LT.0.0001)FLOW=0.0001
C
C INLET AND OUTLET NODES 120 TO 60 AND 104 TO 121 AND 121 TO 120
C
      G(171)=FLOW*13.0
      G(172)=G(171)
      G(173)=G(171)
```

```
C
C 60 TO 59 AND 103 TO 104
C
        G(175)=FLOW*5.0
        G(223)=G(175)
C
C  59 TO 58 AND 102 TO 103
C
        G(177)=FLOW*3.0
        G(222)=G(177)
C
C 58 TO 57 AND 101 TO 102
C
        G(179)=FLOW
        G(221)=FLOW
C
C 57 TO 61. . . ETC.   97 TO 101
C
        G(180)=FLOW
        DO 300 I=181,217,4
300     G(I)=FLOW
C
C 58 TO 62. . . ETC.   98 TO 102
C
        AFLOW=FLOW*2.0
        DO 305 I=178,218,4
305     G(I)=AFLOW
C
C 59 TO 63. . . ETC.   99 TO 103
C
        G(176)=AFLOW
        DO 310 I=183,219,4
310     G(I)=AFLOW
C
C 60 TO 64. . . ETC.   100 TO 104
C.
```

```
      AFLOW=1.5*FLOW
      G(174)=AFLOW
      DO 315 I=184,220,4
315   G(I)=AFLOW
      RETURN
      END
C
      SUBROUTINE STTUBK(FLORAT)
C
C PURPOSE:       THIS SUBROUTINE SETS THE PLATE TO TUBE CONDUCTANCES.
C                SUBROUTINE TUBCON IS CALLED TO ACTUALLY SET THE
C                CONDUCTANCES BASED ON FLUID PROPERTIES, TUBE DIMENSIONS
C                ,AND FLOW RATE.
C
C INPUTS:        FLOW-R- THE OVERALL FLOW RATE FOR THE ENTIRE COLLECTOR.
C
C OUTPUTS:       ALL OUTPUTS ARE TO THE COND COMMON BLOCK.
C
C SUBROUTINES REFERNCED:         TUBCON
C
C RESTRICTIONS: CONDUCTOR NUMBERS ARE REFERENCED BY THEIR POSITION
C               RELATIVE TO THE START OF THE COND COMMON BLOCK.  ANY
C               CHANGES TO THE CONDUCTOR BLOCK MAY REQUIRE THIS
C               SUBROUTINE TO BE UPDATED.
C
C START
C
      COMMON/COND/G(1)
C
C FLOW IS ACTUAL FLOW RATE (LBS/HOUR) 13=NUMBER OF TUBES
C
      FLOW=FLORAT/13.0
C CALCULATE TUBE CONDUCTANCES
C
C CALCULATE END NODES FIRST USING INDIVIDUAL LENGTHS AND
C FLOW RATES
```

```
C
C 8-60 AND 52-104
C
      CALL TUBCON(FLOW*2.0,4.453,HC1)
      CALL TUBCON(FLOW*6.0,2.625,HC2)
      CALL TUBCON(FLOW*5.0,1.250,HC3)
      CALL TUBCON(FLOW,3.750,HC4)
      G(227)=HC1+HC2+HC3+HC4
      G(271)=G(227)
C
C 7-59 AND 51-103
C
      CALL TUBCON(FLOW*5.0,1.250,HC1)
      CALL TUBCON(FLOW,3.063+2.375,HC2)
      CALL TUBCON(FLOW*4.0,2.563,HC3)
      CALL TUBCON(FLOW*3.0,1.250,HC4)
      G(226)=HC1+HC2+HC3+HC4
      G(270)=G(226)
C
C 6-58 AND 50-102
C
      CALL TUBCON(FLOW*3.0,1.250,HC1)
      CALL TUBCON(FLOW,1.688+1.063,HC2)
      CALL TUBCON(FLOW*2.0,2.563,HC3)
      CALL TUBCON(FLOW,1.250,HC4)
      G(225)=HC1+HC2+HC3+HC4
      G(269)=G(225)
C
C 5-59 AND 49-101
C
      CALL TUBCON(FLOW,1.25+0.375,HC1)
      G(224)=HC1
      G(268)=HC1
C
C 12-64 TO 48-100
C
```

```
      CALL TUBCON(FLOW*1.5,6.078,HC1)
      DO 500 I=231,267,4
500   G(I)=HC1
C
C 11-63 TO 47-99 AND 10-62 TO 46-98
C
      CALL TUBCON(FLOW*2.0,6.078,HC1)
      DO 510 I=229,265,4
      G(I)=HC1
510   G(I+1)=HC1
C
C 9-61 TO 45-96
C
      CALL TUBCON(FLOW,6.078,HC1)
      DO 520 I=228,264,4
520   G(I)=HC1
C
      RETURN
      END
C
      SUBROUTINE STFLUX(FLUX)
C
C PURPOSE:       THIS SUBROUTINE SETS THE ABSORBED SOLAR FLUX
C                FOR ALL NODES ON THE COLLECTOR PLATE.
C
C INPUTS:        FLUX-R-SOLAR FLUX (BTU/HR-SQ FT)
C
C OUTPUTS:       ALL OUTPUTS ARE TO THE COND COMMON BLOCK.
C
C SUBROUTINES REFERENCED:        NONE.
C
C RESTRICTIONS: CURRENTLY SETS ONLY THE HEATING RATES FOR THE COLLECTOR
C                NODES.
C
C START
C
```

```
      COMMON/HEAT/Q(1)
      COMMON/ARRAYS/A(1)
C
C SET HEATING RATES FOR ALL NODES
C
      DO 100 I=1,56
100   Q(I)=FLUX*A(I)*0.95
C
C ZERO ALL OTHER HEATING RATES
C
      DO 110 I=57,124
110   Q(I)=0.0
      RETURN
      END
C
      SUBROUTINE ARYDIV(N,ARRY1,VALUE,ARRY2)
C
C PURPOSE:      THIS SUBROUTINE DIVIDES N ELEMNTS OF ARRY1
C               BY VALUE AND STORES THE RESULTS IN ARRY2.
C
C INPUTS:       N-I-NUMBER OF ELEMENTS IN ARRY1 THAT ARE TOBE OPERATED
C                   ON.
C               ARRY1-R-THE ARRAY WHICH IS TO BE OPERATED ON.
C               VALUE-R-THE VALUE WHICH IS DIVIDED INTO THE ELEMENTS
C                       OF ARRY1.
C
C OUTPUTS:      ARRY2-R-THE ARRAY WHICH IS TO RECIEVE THE RESULTS.
C
C SUBROUTINES REFERENCED:       NONE.
C
C RESTRICTIONS: NONE.
C
C START
C
      DIMENSION ARRY1(1),ARRY2(1)
      DO 100 I=1,N
```

```
100     ARRY2(I)=ARRY1(1)/VALUE
        RETURN
        END
C
        SUBROUTINE STRADK
C
C PURPOSE:      THIS SUBROUTINE SETS THE RADIATION CONDUCTORS BETWEEN
C               THE COLLECTOR PLATE AND GLASS PLATES, GLASS TO GLASS
C               PLATES, AND GLASS TO SKY.
C
C INPUTS:       NONE
C
C OUTPUTS:      ALL OUTPUTS ARE TO THE COND COMMON BLOCK.
C
C SUBROUTINES REFERENCED:       NONE.
C
C RESTRICTIONS: CONDUCTOR NUMBERS ARE REFERENCED BY THEIR  POSITION
C               RELATIVE TO THE START OF THE COND COMMON BLOCK.  ANY
C               CHANGES TO THE CONDUCTOR BLOCK MAY REQUIRE THIS
C               SUBROUTINE TO BE UPDATED.
C
C START
C
        COMMON/ARRAYS/A(1)
        COMMON/NODEA/NA(1)
        COMMON/NODEB/NB(1)
        COMMON/COND/G(1)
C
C SET ALL THE RADIATION CONDUCTORS
C
C FIRST SET THE MATERIAL PROPERTIES
C
        EBLKPT=0.95
        EGLSPL=0.95
        F12=1.0
        RHO1=1.0-EBLKPT
```

```
      RHO2=1.0-EGLSPL
C
      DO 600 I=343,410
      INA=NA(I)
      INB=NB(I)
      SUMRES=RHO1/(EBLKPT*A(INA))+1.0/(A(INA)*F12)+
     1   RHO2/(EGLSPL*A(INB))
600   G(I)=-0.1712E-08/SUMRES
      DO 610 I=411,413
      INA=NA(I)
      SUMRES=RHO2/(EGLSPL*A(INA))+1.0/A(INA)
610   G(I)=-0.1712E-08/SUMRES
      RETURN
      END
C
      SUBROUTINE TUBCON(FLOW,TULNTH,CONDUC)
C
C PURPOSE:      THIS SUBROUTINE CALCULATES THE FLUID TO TUBE
C               CONDUCTANCES.  EQUATION 4.14.3 IS USED TO CALCULATE
C               REYNOLDS AND NUSSELT NUMBERS  WHICH ARE USED TO FIND
C               THE CONDUCTANCE VALUE.
C
C INPUTS:       FLOW - FLOW RATE FOR THE TUBE (LBM/HOUR)
C               TULNTH - TUBE LENGTH (INCHES)
C
C OUTPUTS:      CONDUC - CONDUCTANCE BETWEEN FLUID AND TUBE WALL
C                        (BTU/HR-F)
C
C SUBROUTINES REFERENCED:       NONE
C
C RESTRICTIONS: LAMINAR FLOW IS ASSUMED.
C
C START
C
C THE FOLLOWING FLUID PROPERTIES ARE USED:
C       DENSITY=62.4 FLUID DENSITY (LBM/CUB.FT)
```

```
C        TBDIAM=0.440 TUBE DIAMETER (INCHES)
C        AKFLUD=0.364 FLUID CONDUCTIVITY (BTU/HR-SQ.FT-F)
C        DVISC=0.740E-05 KINEMATIC VISCOSITY (SQ.FT/SEC)
C        AVISC=0.458E-03 ABSOLUTE VISCOSITY (LBM/FT-SEC)
C        CPFLUD=1.0 FLUID HEAT CAPACITY (BTU/LBM-F)
C
C RFLOW=FLOW/3600.0
C TBLNTH=TULNTH/12.0
C
         RFLOW=FLOW*2.777777E-04
         TBLNTH=TULNTH*0.0833333
C
C CALCULATE TUBE AREA (SQ FT)
C TBAREA=3.1415926*TBDIAM*TBLNTH
C
         TBAREA=0.11519173*TBLNTH
C
C CHECK FOR STATIC FLOW CONDITIONS
C
C CONSIDER ANY FLOW RATE .LT. 10 LBS/HOUR (=0.769 LBS/HOUR PER TUBE)
C TO HAVE A STATIC FLOW CONDUCTANCE.  NUSSELT CALCULATIONS
C USED IN THIS SUBROUTINE DON'T WORK FOR EXTREMELY LOW FLOW RATES.
C
         IF(FLOW.GT.0.769)GO TO 500
C
C SET STATIC FLOW CONDUCTANCE
C CONDUC=TBAREA*AKFLUD/(TBDIAM/4.0)
C
         CONDUC=TBAREA*39.70909
         GO TO 9999
C
C CALCULATE THE REYNOLDS NUMBER
C RENOLD=(RFLOW*TBDIAM)/(3.1415926*TBDIAM*TBDIAM*AVISC/4.0)
C
500      RENOLD=7.581815E04*RFLOW
C
```

```
C CALCULATE THE PRANDTL NUMBER
C PRNDTL=CPFLUD*AVISC/AKFLUD
C PRNDTL=4.529670
C
C CALCULATE THE NUSSELT NUMBER
C SCR1=ESQRT(RENOLD*PRNDTL*TBDIAM/TBLNTH)
C
      SCR1=ESQRT(RENOLD*166.0879E-03/TBLNTH)
C
C SCR2=PRNTDL**0.167
C SCR2=1.2869543
C
      SCR10=1.0/(1.0-(2.654/(1.2869543*SCR1)))
      SCR3=ELOG(SCR10)
C
C SCR4=(RENOLD*PRNDTL*TBDIAM)/(4*TBLNTH)
C
      SCR4=41.521975E-03*RENOLD/TBLNTH
      ANUSLT=SCR3*SCR4
C
C CALCULATE THE CONVECTION COEFFICIENT
C HC=(ANUSLT*AKFLUD/TBDIAM)*3600.0
C
      HC=ANUSLT*9.9272727
C
C CALCULATE THE CONDUCTANCE VALUE
C
      CONDUC=HC*TBAREA
C
9999  RETURN
      END
C
      SUBROUTINE CONVRS
C
C PURPOSE:    TO CONVERT INPUT BOUNDARY DATA TO ENGLISH ENGINEERING
C             UNITS.  (THE THERMAL ANALYZER PROGRAM WAS DESIGNED USING
```

```
C                ENGLISH ENGINEERING UNITS).
C
C INPUTS:        ALL INPUTS ARE VIA THE COMMON BLOCK BNDRY.
C
C OUTPUTS:       ALL OUTPUTS ARE VIA THE COMMON BLOCK BNDRY.
C
C SUBROUTINE REFERENCED:         NONE.
C
C RESTRICTIONS: NONE.
C
C START
C
      DIMENSION TABLE1(16),TABLE2(16)
      COMMON/BNDRY/NDATA1,NDATA2,TIMHR1(50),TIMHR2(50),SOLAR(50),
     &             TMPAMB(50),WNDS(50),WNDD(50)
      DATA TABLE1/'N   ','NNE ','NE  ','ENE ','E   ','ESE ','SE  ',
     &            'SSE ','S   ','SSW ','SW  ','WSW ','W   ','WNW ',
     &            'NW  ','NNW '/
      DATA TABLE2/0.0,0.392699,0.785398,1.1780972,1.5707963,
     &            1.9634953,2.3561944,2.7488935,3.1415926,3.5342917,
     &            3.9269907,4.3196898,4.7123889,5.105088,5.497787,
     &            5.8905861/
C
C CONVERT SOLAR INSOLATION FROM WATTS/SQ.M TO BTU/HR-SQFT
C
      DO 10 I=1,NDATA1
10    SOLAR(I)=SOLAR(I)/3.17210
C
C CONVERT AMBIENT CELSIUS TO FAHRENHEIT
C
      DO 20 I=1,NDATA2
20    TMPAMB(I)=TMPAMB(I)*1.8+32.0
C
C CONVERT WINDSPEED FROM KM/HR TO MILES PER HOUR
C
      DO 30 I=1,NDATA2
```

```
30      WNDS(I)=WNDS(I)*0.621371
C
C CONVERT ALPHABETIC WIND DIRECTION TO RADIANS
C
        DO 40 I=1,NDATA2
C
C STEP THROUGH THE LOOK UP TABLE
C
        DO 50 J=1,16
50      IF(WNDD(I).EQ.TABLE1(J))GO TO 60
        WRITE(1,1000)
1000    FORMAT(1H ,'***ERROR - NO MATCH IN LOOK UP TABLE IN ',
     &           'SUBROUTINE CONVRS')
60      WNDD(I)=TABLE2(J)
40      CONTINUE
        RETURN
        END
```

FILE: FTN4UTIL.REL

```
      SUBROUTINE DECBA6(TIME,HOURS,MINS,SECS,TIME6)
C
C PURPOSE:      TO CONVERT DECIMAL TIME (HHH.FFFF) TO BASE 6 TIME
C               (HHH.MMSS).
C
C INPUTS:       TIME-R-CONTAINS TIME IN DECIMAL TO BE CONVERTED TO
C               BASE 6.
C
C OUTPUTS:      HOURS-I-HOURS
C               MINS-I-MINUTES
C               SECS-I-SECONDS
C               TIME6-R-BASE 6 TIME (HHH.MMSS)
C
C SUBROUTINES CALLED:   NONE
C
C RESTRICTIONS: NONE
C
C START
      INTEGER HOURS,MINS,SECS
C FORM HOURS,MINUTES,SECONDS
      HOURS=IFIX(TIME)
C GET FRACTIONAL PART WHICH CONTAINS MINUTES AND SECONDS
      FRAC1=TIME-HOURS
C GET AN INTERMEDIATE VALUE
      DUM1=FRAC1*60.0
C CALCULATE MINUTES
      MINS=IFIX(DUM1)
C GET THE FRACTIONAL PART WHICH CONTAINS SECONDS
      FRAC2=DUM1-MINS
C CALCULATE SECONDS. THE 0.5 TAKES CARE OF ROUND OFF ERROR
      FRAC3=FRAC2*60.0+0.5
      SECS=IFIX(FRAC3)
      IF(SECS.NE.60)GO TO 10
      SECS=0
      MINS=MINS+1
C FORM TIME6
10    TIME6=HOURS+(MINS*1E-02)+(SECS*1E-04)
      RETURN
      END
```

```
      SUBROUTINE BA6DEC(TIME,HOURS,MINS,SECS,DTIME)
C
C PURPOSE:       TO CONVERT TIME (BASE 6 OF FORM HHH.MMSS) TO DECIMAL
C                (BASE 10 OF FORM HHH.FFFF) WHERE F=FRACTIONAL.
C
C INPUTS:        TIME-R-CONTAINS TIME IN BASE 6 TO BE CONVERTED TO
C                DECIMAL.
C
C OUTPUTS:       HOURS-I-HOURS
C                MINS-I-MINUTES
C                SECS-I-SECONDS
C                DTIME-R-DECIMAL TIME (HHH.FFFF)
C
C SUBROUTINES CALLED:   NONE
C
C RESTRICTIONS: NONE
C
C START
      INTEGER HOURS,MINS,SECS
C SEPARATE HOURS,MINUTES,SECONDS
      HOURS=IFIX(TIME)
C DEFINE AN INTERMEDIATE VALUE
      DUM1=(TIME-HOURS)*100.0
C CALCULATE MINS.
      MINS=IFIX(DUM1)
C CALCULATE SECONDS.
      DUM2=(DUM1-MINS)*100.0+0.5
      SECS=IFIX(DUM2)
C CONVERT TO DECIMAL
      DTIME=HOURS+(MINS/60.0)+(SECS/3600.0)
      RETURN
      END
C
      SUBROUTINE SKIP(IUNIT,LINES)
C
```

```
C PURPOSE:       TO WRITE BLANK LINES ON IUNIT
C
C INPUT:         IUNIT-I-A-UNIT NUMBER TO WRITE TO.
C                LINES-I-A-NUMBER OF BLANK LINES TO WRITE.
C
C OUTPUT:        BLANK LINES TO IUNIT
C
C SUBROUTINES REFERENCED:        NONE
C
C RESTRICTIONS: NONE
C
C INITIALIZE
C
         DO 10 I=1,LINES
10       WRITE(IUNIT,1000)
1000     FORMAT(/)
         RETURN
         END
C
         SUBROUTINE UNPACK(IFILE,IBUF2,FIRST)
C
C PURPOSE:       TO UNPACK AN EDIT FORMAT FILE BY PLACING INTO A
C                LOGICAL ARRAY (IBUF2) A SINGLE FORTRAN FORMATTED
C                RECORD FROM IFILE TERMINATED BY A CARRIAGE RETURN.
C
C INPUTS:        IFILE-THE EDIT FORMAT FILE TO READ FROM.
C                IBUF2-THE LOGICAL ARRAY (SIZE=128) TO RECIEVE A
C                      FORTRAN FORMATTED RECORD.
C                FIRST-FLAG TO INDICATE IF THIS IS THE FIRST TIME
C                      THIS ROUTINE HAS BEEN CALLED.
C
C OUTPUTS:       IBUF2-CONTAINS THE FORTRAN FORMATTED RECORD.
C
C SUBROUTINES REFERENCED:        NONE.
C
C RESTRICTIONS: RANDOM ACCESS TO RECORDS IN THE FORMAT FILE IS NOT
```

```
C                ALLOWED.  RECORDS MUST BE ACCESSED VIA A SEQUENTIAL
C                READ FROM THE BEGINNING OF THE FILE.  REWIND THE
C                FILE TO START OVER AGAIN.  THE FIRST TIME THIS
C                ROUTINE IS CALLED FIRST=.TRUE..  FIRST MUST NOT
C                BE CHANGED BY ANY OTHER ROUTINE.
C
C INITIALIZE
C
C
      LOGICAL IBUF1(128),IBUF2(128),FIRST
C
C BLANK FILL IBUF2
C
      DO 10 J=1,128
10    IBUF2(J)=Z'20'
C
      J=0
      IF(.NOT.FIRST)GO TO 200
C
100   READ(IFILE)(IBUF1(IJ),IJ=1,128)
      I=0
200   I=I+1
      IF(I.GT.128)GO TO 100
C
C IGNORE LINE FEEDS
C
      IF(IBUF1(I).EQ.Z'0A')GO TO 200
C
C OTHERWISE TRANSFER TO IBUF2
C
      J=J+1
      IBUF2(J)=IBUF1(I)
C
C RETURN IF A CARRIAGE RETURN
C
      IF(IBUF1(I).EQ.Z'0D')GO TO 9000
```

```
      GO TO 200
C
9000  FIRST=.FALSE.
      RETURN
      END
C
      FUNCTION AINGRL(X,Y,B,C,D,N,I,X1,J,X2)
C
C PURPOSE:      THIS SUBPROGRAM INTEGRATES A SET OF DATA WHICH HAS
C               HAD IT'S DISCRETE CUBIC SPLINE EQUATION COEFFICIENTS
C               PREVIOUSLY CALCULATED BY SUBROUTINE DCSINT AND STORED
C               IN ARRAYS B,C,D.
C
C INPUTS:       X - THE ARRAY OF INDEPENDENT VALUES.
C               Y - THE ARRAY OF DEPENDENT VALUES.
C               B,C,D - ARRAYS OF COEFFICIENTS CALCULATED BY
C                       SUBROUTINE DCSINT.
C               N - THE NUMBER OF DATA POINTS I.E. THE DIMENSION
C                       OF ARRAYS X,B,C,D.
C               I = 0 - THIS ROUTINE IS TO FIND THE INTERVAL THAT
C                       X1 LIES IN.
C                 > 0 - SPECIFIES THE INTERVAL IN WHICH X1 IS TO BE
C                       FOUND.
C               X1 - THE UPPER LIMIT OF THE INTEGRAL WHICH STARTS AT
C                       0.0 AND ENDS AT X1.
C               J = 0 - THIS ROUTINE IS TO FIND THE INTERVAL THAT X2
C                       LIES IN.
C                 > 0 - SPECIFIES THE INTERVAL IN WHICH X2 IS TO BE
C                       FOUND.
C               X2 - THE UPPER LIMIT OF THE INTEGRAL WHICH STARTS
C                       AT 0.0 AND ENDS AT X1.
C
C OUTPUTS:      AINGRL - CONTAINS THE INTEGRAL FROM X1 TO X2.
C               I,J - CONTAIN THE INTERVALS IN WHICH X1 AND X2
C                       RESPECTIVELY ARE FOUND.
C
```

```
C SUBROUTINE REFERENCED:        SPLINT
C
C RESTRICTIONS: IT IS ASSUMED THAT X(I).LT.X(I+1) ALWAYS.
C               ALSO X(J).GT.X(I)
C
C START
C
      DIMENSION X(1),Y(1),B(1),C(1),D(1)
C
C FIND THE INTERVAL IF I=0
C
      IF(I.NE.0) GO TO 40
      DO 20 I=1,N
20    IF(X1.LT.X(I))GO TO 30
30    I=I-1
C
C FIND THE INTERVAL IF J=0
C
40    IF(J.NE.0)GO TO 70
      DO 50 J=1,N
50    IF(X2.LT.X(J))GO TO 60
60    J=J-1
C
C BOTH INTERVALS KNOWN;  CHECK FOR THE 3 CASES
C
70    CONTINUE
C
C CASE 1:  BOTH X1 AND X2 LIE IN THE SAME INTERVAL (I.E. I=J)
C
      IF(I.NE.J)GO TO 500
      AINGRL=SPLINT(X,Y,B,C,D,N,J,X2)-SPLINT(X,Y,B,C,D,N,I,X1)
      RETURN
C
C CASE2:  X1 AND X2 LIE IN ADJACENT INTERVALS (I.E. I=J-1)
C
500   IF(I.NE.(J-1))GO TO 510
```

```
      SUM1=SPLINT(X,Y,B,C,D,N,I,X(J))-SPLINT(X,Y,B,C,D,N,I,X1)
      SUM2=SPLINT(X,Y,B,C,D,N,J,X2)
      AINGRL=SUM1+SUM2
      RETURN
C
C CASE 3:  X1 AND X2 LIE IN INTERVALS SEPARATED BY ONE OR MORE INTERVALS
C
510   SUM1=SPLINT(X,Y,B,C,D,N,I,X(I+1))-SPLINT(X,Y,B,C,D,N,I,X1)
      SUM2=SPLINT(X,Y,B,C,D,N,J,X2)
      ITRVL1=I+1
      ITRVL2=J-1
      SUM3=0.0
      DO 520 L=ITRVL1,ITRVL2
520   SUM3=SUM3+SPLINT(X,Y,B,C,D,N,L,X(L+1))
      AINGRL=SUM1+SUM2+SUM3
      RETURN
      END
C
      FUNCTION SPLEVL(X,Y,B,C,D,N,I,X1)
C
C PURPOSE:      THIS SUBPROGRAM EVALUATES THE DISCRETE CUBIC SPLINE
C               INTERPOLATION:
C                       SPLEVL=Y(I)+B(I)*DELTA+C(I)*DELTA**2
C                                +D(I)*DELTA**3
C               WHERE DELTA=X1-X(I).  THE CALLING PROGRAM MAY REQUEST
C               THAT A SEARCH BE MADE FIRST TO DETERMINE THE INTERVAL,
C               I, WHICH SATISFIES THE FOLLOWING:
C                       X(I).LT.X1.LT.X(I+1)
C               THIS FUNCTION IS DESIGNED TO BE USED WITH SUBROUTINE
C               DCSINT.
C
C INPUTS:       X - THE ARRAY OF INDEPENDENT VALUES
C               Y - THE ARRAY OF DEPENDENT VALUES
C               B,C,D - ARRAYS OF COEFFICIENTS PRODUCED BY SUBROUTINE
C                       DCSINT.
C               N - NUMBER OF DATA POINTS
```

```
C                I = 0 - THIS SUBPROGRAM MUST FIND THE INTERVAL FOR WHICH
C                      X(I).LT.X1.LT.X(I+1)
C                  > 0 - THE CALLING PROGRAM HAS FOUND I AND SUPPLIED IT
C                      AS I.
C                X1 - VALUE FOR WHICH AN INTERPOLATED Y VALUE IS TO
C                      BE FOUND.
C
C OUTPUTS:       I - CONTAINS THE INTERVAL FOR WHICH X(I).LT.X1.LT.X(I+!)
C                SPLEVL - CONTAINS THE INTERPOLATED RESULT
C
C SUBROUTINES REFERENCED:        NONE
C
C RESTRICTIONS: IT IS ASSUMED THAT THE X ARRAY VALUES SATISFY THE
C                X(I).LT.X(I+1)
C
C START
C
      DIMENSION X(1),Y(1),B(1),C(1),D(1)
C
C FIND THE INTERVAL IF I=0
C
      IF(I.NE.0)GO TO 500
      DO 10 J=1,N
10    IF(X1.LT.X(J))GO TO 20
C
20    I=J-1
C
500   DELTA=X1-X(I)
C
C EVALUATE THE FUNCTION
C
      SPLEVL=Y(I)+(B(I)+(C(I)+D(I)*DELTA)*DELTA)*DELTA
C
      RETURN
      END
C
```

```
      FUNCTION SPLINT(X,Y,B,C,D,N,I,X1)
C
C PURPOSE:      THIS SUBPROGRAM GIVES THE INTEGRAL OVER AN INTERVAL
C               DEFINED FROM X=0.0 TO X=X1 FOR THE DISCRETE CUBIC
C               SPLINE INTERPOLATION EQUATION.  A SEARCH TO DETERMINE
C               THE INTERVAL THAT X1 LIES IN CAN BE MADE BY SETTING
C               I=0.  THE EQUATION SOLVED FOR IS:
C               SPLINT=(Y(I)+(B(I)/2.0+(C(I)/3.0+(D(I)/4.0*DELTA))
C                     *DELTA)*DELTA)*DELTA
C               THIS FUNCTION IS DESIGNED TO BE USED WITH SUBROUTINE
C               DCSINT.
C
C INPUTS:       X - THE ARRAY OF INDEPENDENT VARIABLES
C               Y - THE ARRAY OF DEPENDENT VARIABLES
C               B,C,D - ARRAYS OF COEFFICIENTS CALCULATED BY SUBROUTINE
C               DCSINT.
C               N - NUMBER OF ELEMENTS IN THE X ARRAY.
C               I = 0 - THIS SUBPROGRAM MUST FIND THE INTERVAL FOR
C                       WHICH X(I).LT.X1.LT.X(I+1).
C               X1 - UPPER LIMIT OF THE INTEGRATION
C
C OUTPUTS:      I - CONTAINS THE INTERVAL FOR WHICH X(I).LT.X1.LT.(XI=1)
C               SPLINT - CONTAINS THE INTEGRAL FROM 0.0 TO (X1-X(I))
C
C SUBROUTINES REFERENCED:       NONE.
C
C RESTRICTIONS: IT IS ASSUMED THAT THE X ARRAY VALUES SATISFY THE
C               FOLLOWING:  X(I).LT.X(I+1).
C               ALSO, SINCE THE CONSTANT OF INTEGRATION IS NOT KNOWN
C               THIS FUNCTION MAY ONLY BE USED FOR DEFINITE INTEGRALS.
C
C START
C
      DIMENSION X(1),Y(1),B(1),C(1),D(1)
C
C FIND THE INTERVAL IF I=0
```

```
C
      IF(I.NE.0)GO TO 500
      DO 10 J=1,N
10    IF(X1.LT.X(J))GO TO 20
20    I=J-1
C
500   DELTA=X1-X(I)
C
C EVALUATE THE FUNCTION
C
      SPLINT=(Y(I)+(B(I)/2.0+(C(I)/3.0+(D(I)/4.0*DELTA))*DELTA)*
     &       DELTA)*DELTA
C
      RETURN
      END
C
      SUBROUTINE DCSINT(IENT,H,N,TNODE,G,END1,ENDN,B,C,D)
C
C THIS SUBROUTINE COMPUTES THE DISCRETE CUBIC SPLINE
C DEFINED ON THE INTERVAL (TNODE(1),TNODE(N)),WHICH INTER-
C POLATES THE DATA (TNODE(I),G(I)),I=1,2,...,N.  WE REQUIRE
C THAT TNODE(I).LT.TNODE(I+1). END1 AND ENDN CONTAIN THE
C VALUES OF THE END CONDITIONS BEING USED.
C
C IF IENT=1, THE FIRST CENTRAL DIVIDED DIFFERENCE END
C CONDITIONS ARE BEING USED.
C
C IF IENT=2,THE SECOND CENTRAL DIVIDED DIFFERENCE END
C CONDITIONS ARE BEING USED.
C
C IF IENT=3, THE PERIODIC END CONDITIONS ARE BEING USED.
C FOR THIS CASE THE CONTENTS OF G(N), END1,AND ENDN ARE
C IGNORED.
C
C FOR ALL THREE END CONDITIONS N MUST BE GREATER THAN OR
C EQUAL TO 2.
```

```
C
C THE DISCRETE CUBIC SPLINE IS REPRESENTED BY PIECEWISE
C CUBIC POLYNOMIALS. FOR T IN THE INTERNAL (TNODE(I),TNODE
C (I+1)) THE CUBIC SPLINE IS
C
C    S(T)=G(I)+B(I)*(T-TNODE(I))+C(I)*(T-TNODE(I))**+
C                D(I)*(T-TNODE(I))**3
C
      DIMENSION TNODE(N),G(N),B(N),C(N),D(N)
C
C INPUT PARAMETERS (NONE OF THESE PARAMETERS
C                          ARE CHANGED BY THIS SUBROUTINE.)
C
C IENT - SPECIFIES END CONDITONS WHICH ARE IN EFFECT.
C H - THE STEP SIZE USED FOR THE DISCRETE CUBIC SPLINE.
C N - NUMBER OF NODES (TNODE) AND DATA VALUES (G).
C     (N.GE.2)
C TNODE - REAL ARRAY CONTAINING THE NODES (TNODE(I).LT.TNODE(I+1)).
CG - REAL ARRAY CONTAINING THE INTERPOLATING DATA.
C END1 - END CONDITION VALUE AT TNODE(1).
C ENDN - END CONDITION VALUE AT TNODE(N).
C
C OUTPUT PARAMETERS
C
C B - REAL ARRAY CONTAINING COEFFICIENTS OF
C               (T-TNODE(I)),I=1,2,...,N-1.
C C - REAL ARRAY CONTAINING COEFFICIENTS OF
C               (T-TNODE(I))**2,I=1,2,...,N-1.
C
C D - REAL ARRAY CONTAINING COEFFICIENTS OF
C               (T-TNODE(I))**3,I=1,2,...,N-1.
C
C START
C
C SPECIAL CASES ARE ACCOUNTED FOR HERE.
      IF((N.EQ.2).AND.(IENT.EQ.3))GO TO 220
```

```
      H2=H*H
      N1=N-1
      IF((N.EQ.2).AND.(IENT.EQ.2))GO TO 180
      N2=N1-1
C THE SYMMETRIC TRIDIAGONAL (OR NEAR TRIDIAGONAL) LINEAR
C SYSTEM WILL NOW BE SET UP FOR THE APPROPRIATE END
C CONDITIONS
      HI=TNODE(2)-TNODE(1)
      H2DHI=H2/HI
      ETA2=HI+HI+H2DHI
      GO TO (10,40,60),IENT
C IENT=1 - FIRST CENTRAL DIVIDED DIFFERENCE END CONDITIONS
C SET UP.
10    B(1)=ETA2
      D(1)=HI-H2DHI
      G2=(G(2)-G(1))/HI
      C(1)=3.0*(G2-END1)
      IF(N.EQ.2)GO TO 30
      DO 20 I=2,N1
            ETA1=ETA2
            G1=G2
            HI=TNODE(I+1)-TNODE(I)
            H2DHI=H2/HI
            ETA2=HI+HI+H2DHI
            B(I)=ETA1+ETA2
            D(I)=HI-H2DHI
            G2=(G(I+1)-G(I))/HI
            C(I)=3.0*(G2-G1)
20    CONTINUE
30    B(N)=ETA2
      C(N)=3.0*(ENDN-G2)
      L=N
C SET UP FOR (1) FIRST CENTRAL DIVIDED DIFFERENCE END
C CONDITIONING COMPLETE.  THE LINEAR EQUATIONS WILL BE NOW
C SOLVED.
      GO TO 110
```

```
C IENT=2 - SECOND CENTRAL DIVIDED DIFFERENCE END CONDITIONS
C SET UP.
40      GAMMA=HI-H2DHI
        G2=(G(2)-G(1))/HI
        DO 50 I=1,N2
                ETA1=ETA2
                G1=G2
                HI=TNODE(I+2)-TNODE(I+1)
                H2DHI=H2/HI
                ETA2=HI+HI+H2DHI
        B(I)=ETA1+ETA2
                D(I)=HI-H2DHI
                G2=(G(I+2)-G(I+1))/HI
                C(I)=3.0*(G2-G1)
50      CONTINUE
        C(1)=C(1)-GAMMA*END1/2.0
        HI=TNODE(N)-TNODE(N1)
        GAMMA=HI-H2/HI
        C(N2)=C(N2)-GAMMA*ENDN/2.0
C STEP UP FOR (2) SECOND CENTRAL DIVIDED DIFFERENCE
C END CONDITIONS COMPLETE. THE LINEAR EQUATIONS WILL NOW
C BE SOLVED.
        IF(N2.EQ.1)GO TO 100
        L=N2
        GO TO 110
C IENT=3 - PERIODIC END CONDITIONS SET UP.
60      B(1)=ETA2
        D(1)=HI-H2DHI
        DO 70 I=2,N1
                ETA1=ETA2
                HI=TNODE(I+1)-TNODE(I)
                H2DHI=H2/HI
                ETA2=HI+HI+H2DHI
                B(I)=ETA1+ETA2
                D(I)=HI-H2DHI
                C(I)=0.0
```

```
70      CONTINUE
        CT=D(N1)
        C(1)=CT
        C(N1)=CT
        B(1)=B(1)+ETA2-CT
        B(N1)=B(N1)-CT
        L=N1
        ITRANS=1
        GO TO 120
80      G1=(G(N1)-G(N2))/(TNODE(N1)-TNODE(N2))
        G2=(G(1)-G(N1))/(TNODE(N)-TNODE(N1))
        DEN=(1.0+C(1)+C(N1))
        CT=3.0*(G2-G1)
        BS1=CT*C(N1)
        C(N1)=CT
        DO 90 I=1,N2
                HI=TNODE(I+1)-TNODE(I)
                G1=G2
                G2=(G(I+1)-G(I))/HI
                CT=3.0*(G2-G1)
                BS1=BS1+CT*C(I)
                C(I)=CT
90      CONTINUE
        BS1=BS1/DEN
        C(1)=C(1)-BS1
        C(N1)=C(N1)-BS1
        ITRANS=0
        GO TO 140
C THE SET UP AND MOST OF THE DETAILS FOR SOLVING THE
C LINEAR EQUATION FOR (3) THE PERIODIC END CONDITIONS
C ARE COMPLETED.
C
C THE LINEAR EQUATION ARE SOLVED HERE.
100     C(2)=C(1)/B(1)
        GO TO 180
110     ITRANS=0
```

```
120     L1=L-1
        DO 130 I=1,L1
                T=D(I)/B(I)
                B(I+1)=B(I+1)-T*D(I)
                D(I)=T
130     CONTINUE
C THE LINEAR EQUATION SOLVER IS ENTERED AT THIS POINT
C IF THE LU FACTORIZATION HAS ALREADY BEEN DONE.
140     DO 150 I=1,L1
                C(I+1)=C(I+1)-D(I)*C(I)
150     CONTINUE
        C(L)=C(L)/B(L)
        DO 160 I=1,L1
                LI=L-I
                C(LI)=C(LI)/B(LI)-D(LI)*C(LI+1)
160     CONTINUE
        IF(ITRANS.GE.1)GO TO 80
C THE LINEAR SYSTEM HAS BEEN SOLVE FOR THE C-VECTOR
        IF(IENT.EQ.3)C(N)=C(1)
        IF(IENT.NE.2)GO TO 190
        DO 170 I=1,N2
                LI=N-I
        C(LI)=C(LI-1)
170     CONTINUE
180     C(1)=END1/2.0
        C(N)=ENDN/2.0
190     C1=C(1)
        C2=C(2)
        HI=TNODE(2)-TNODE(1)
        IF(N.EQ.2)GO TO 210
        DO 200 I=1,N2
                B(I)=(G(I+1)-G(I))/HI-HI*(C1+C1+C2)/3.0
                D(I)=(C2-C1)/(3.0*HI)
                HI=TNODE(I+2)-TNODE(I+1)
                C1=C2
                C2=C(I+2)
```

```
200     CONTINUE
210     GN=G(1)
        IF(IENT.NE.3)GN=G(N)
        B(N1)=(GN-G(N1))/HI-HI*(C1+1+C2)/3.0
        D(N1)=(C2-C1)/(3.0*HI)
C THE INTERPOLATING DISCRETE CUBIC SPLINE HAS BEEN
C CONSTRUCTED.
        RETURN
C THE FOLLOWING HANDLES THE TRIVIAL PERIODIC CASE
C (IENT.EQ.3) WHEN N.EQ.2.
220     B(1)=0.0
        C(1)=0.0
        D(1)=0.0
        RETURN
        END
C
        SUBROUTINE GTTIME(IWKDAY,IDAY,IMON,IYR,IHR,IMIN,ISEC)
C
C PURPOSE:      PROVIDES DAY, MONTH, YEAR, HOURS, MINUTES, SECONDS
C               AND DAY OF WEEK FROM THE REAL TIME CLOCK.
C
C INPUTS:       NONE.
C
C OUTPUTS:      IWKDAY - DAY OF WEEK
C               IDAY,IMON,IYR - DAY, MONTH, AND YEAR IN INTEGER FORMAT
C               IHR,IMIN,ISEC - HOURS, MINUTES, AND SECONDS IN 24 HOUR
C                       INTEGER FORMAT.
C
C SUBROUTINES REFERENCED:       NONE.
C
C RESTRICTIONS: CLOCK COMMAND PORT IS AT 5AH
C               CLOCK DATA PORT IS AT 5BH
C
C START
C
        INTEGER*1 C,D
```

```
      DATA C,D/Z'5A',Z'5B'/
C
C COMMAND A READ OPERATION; FREEZE CLOCK WHILE READING.
C
      CALL OUT(C,Z'50')
      ISEC=INP(D)
      CALL OUT(C,Z'51')
      ISEC=ISEC+INP(D)*10
C
      CALL OUT(C,Z'52')
      IMIN=INP(D)
      CALL OUT(C,Z'53')
      IMIN=IMIN+INP(D)*10
C
      CALL OUT(C,Z'54')
      IHR=INP(D)
      CALL OUT(C,Z'55')
      IHR=IHR+(INP(D).AND.Z'03')*10
C
      CALL OUT(C,Z'57')
      IDAY=INP(D)
      CALL OUT(C,Z'58')
      IDAY=IDAY+(INP(D).AND.Z'03')*10
C
      CALL OUT(C,Z'59')
      IMON=INP(D)
      CALL OUT(C,Z'5A')
      IMON=IMON+INP(D)*10
C
      CALL OUT(C,Z'5B')
      IYR=INP(D)
      CALL OUT(C,Z'5C')
      IYR=IYR+INP(D)*10
C
      CALL OUT(C,Z'56')
      IWKDAY=INP(D)
```

```
C
      CALL OUT(C,Z'00')
      RETURN
      END
```

'49

APPENDIX II

SEARCH SUBPROGRAM LISTING

This appendix presents a listing, in Z-80 Assembler language, of the SEARCH subprogram. It's function is to perform a binary search for adjoining node numbers when called by subroutine SOLVE. This subprogram is well documented and self explanatory.

```
        TITLE *** SEARCH ROUTINE FOR THE THERMAL ANALYZER PROGRAM ***
        ENTRY   SEARCH
;
; PURPOSE:      THIS PROGRAM IS USED AS A SUBROUTINE BY THE THERMAL
;               ANALYZER PROGRAM.  IT'S FUNCTION IS TO SEARCH THROUGH
;               TWO ADJOINING NODE NUMBERS LOOKING FOR A MATCH WITH
;               NODEID.  WHEN A VALID MATCH IS FOUND THE ROW THAT THE
;               CORRESPONDING CONDUCTOR CAN BE FOUND IN IS STORED IN
;               IROW AND THE ATTACHING NODE IS STORED IN NODE.  MATCHING
;               NEGATIVE NODE NUMBERS ARE INVALID AND THEREFORE SKIPPED.
;               ARRAY NA IS SEARCHED FIRST AND THEN NB IS SEARCHED.
;               IF A MATCH HAS NOT BEEN FOUND AT THE END OF THE NB ARRAY,
;               IROW IS SET TO ZERO AND A RETURN IS EXECUTED.
;
; INPUTS:       INPUTS ARE VIA STANDARD FORTRAN SUBROUTINE ARGUEMENTS
;               AND VIA COMMON BLOCKS.  NOTE THAT THE ACTUAL SIZE OF
;               THE COMMON BLOCKS ARE DECLARED BY THE FORTRAN CALLING
;               ROUTINE.
;               HL - CONTAINS THE ADDRESS OF THE NODEID PARAMETER.
;               DE - CONTAINS THE ADDRESS OF THE IROW PARAMETER.
;               BC - CONTAINS THE ADDRESS OF THE NODE PARAMETER.
;               NA - FORTRAN DEFINED COMMON BLOCK CONTIAINING THE
;                    FIRST SET OF NODE NUMBERS.
;               NB - FORTRAN DEFINED COMMON BLOCK CONTAINING THE SECOND
;                    SET OF NODE NUMBERS.
;               NCOND - CONTAINED IN THE CONST COMMON BLOCK OF THE
;                    FORTRAN MAIN PROGRAM  - USED TO CALCULATE THE
;                    ADDRESS OF THE LAST ELEMENTS IN NA AND NB.
;
; OUTPUTS:      ONLY THE IROW, NODE PARAMETERS AND LAST ELEMENTS IN
;               NA AND NB ARE ALTERED.
;               HL - CONTAINS THE ADDRESS OF THE ;NODEID PARAMETER.
;               DE - CONTAINS THE ADDRESS OF THE IROW PARAMETER.
;               BC - CONTAINS THE ADDRESS OF THE NODE PARAMETER.
;
; SUBROUTINES REFERENCED:       NONE.
```

```
;
; RESTRICTIONS: NCOND MUST HAVE BEEN PREVIOUSLY SET TO THE NUMBER OF
;               CONDUCTORS IN THE ARRAY.  INITIALLY THE LAST NODE NUMBERS
;               IN THE NA AND NB ARRAYS MUST HAVE BEEN SET TO 32767, I.E.
;               BITS 6 TO 0 OF THE HIGH ORDER BYTE OF THE LAST ELEMENT IN
;               THE NA AND NB ARRAYS MUST BE SET.
;
; INITIALIZE
;
        FORM
        REM +++ DEFINE STARTING ADDRESSES FOR NA AND NB AND OTHER CONSTANTS +++
        COM     NODEA
NAADDR: DS      1               ; FORTRAN PROGRAM DEFINES ACTUAL SIZE
        COM     NODEB
NBADDR: DS      1               ; FORTRAN PROGRAM DEFINES ACTUAL SIZE
        COM     CONST
        DS      2               ; SKIP OVER FIRST CONSTANT
NCOND:  DS      2               ; AND DEFINE NCOND ADDRESS AND SIZE
;
        REL
;
NBFLAG: DS      1               ; STORAGE LOCATION FOR THE NB FLAG
NIDADR: DS      2               ; TEMPORARY STORAGE FOR
ROWADR: DS      2               ; PARAMETERS PASSED BY
NODADR: DS      2               ; FORTRAN SUBROUTINE CALL
        FORM
        REM +++ START OF PROGRAM +++
SEARCH: LD      (NIDADR),HL     ; SAVE THE NODEID ADDRESS
        LD      (ROWADR),DE     ; SAVE THE IROW ADDRESS
        LD      (NODADR),BC     ; SAVE THE NODE ADDRESS
        LD      IX,(NIDADR)     ; KEEP WORKING COPIES OF THE
        LD      IY,(ROWADR)     ; FIRST TWO ARGUEMENTS IN IX AND IY
;
; CHECK FOR IROW = 0 I.E. IS THIS THE START OF A NEW PASS ON NA AND NB?
;
        LD      A,(IY+0)        ; GET THE LOW ORDER BYTE OF IROW IN A
```

```
        CP      A,00H           ; IS IT ZERO?
        JP      NZ,IROWNZ       ; NO- GO TO IROW NOT ZERO,
                                ; I.E. THIS IS A CONTINUATION OF A PREVIOUS SEARCH
        LD      A,(IY+1)        ; OTHERWISE CHECK HIGH ORDER BYTE
        CP      A,00H           ; IS IT ZERO?
        JP      NZ,IROWNZ       ; NO - GO TO IROW NOT ZERO, I.E. THIS IS A
                                ; CONTINUATION OF A PREVIOUS SEARCH
;
; AT THIS POINT IROW MUST BE ZERO THEREFORE SET UP TO START SEARCH AT THE TOP OF NA
;
        LD      (NBFLAG),A      ; STORE THE ZEROES ALREADY IN A INTO NBFLAG.
;
; STORE THE NODEID IN THE LAST ELEMENTS OF NA AND NB
;
        LD      A,(IX+0)        ; GET THE LOW ORDER BYTE OF NODEID IN A
        LD      DE,NAADDR       ; GET THE NA START ADDRESS
        LD      HL,(NCOND)      ; GET THE NUMBER OF CONDUCTORS
        ADD     HL,HL           ; DOUBLE IT (2 BYTES PER INTEGER) HL=NCOND*2
        EX      DE,HL           ; SAVE NCOND*2 IN DE
        ADD     HL,DE           ; HL=NAADDR+(NCOND*2) HL POINTS
                                ; TO LOW ORDER BYTE OF LAST ELEMENT IN NA
        LD      (HL),A          ; SAVE THE NODEID
;
; REPEAT THE ABOVE FOR THE NBADDR, NOTE THAT DE CONTAINS NCOND*2
; AFTER THE ABOVE 'EX' INSTRUCTION
;
        LD      HL,NBADDR       ; GET THE NB START ADDRESS
        ADD     HL,DE           ; HL=NBADDR+(NCOND*2) HL POINTS
                                ; TO THE LOW ORDER BYTE OF THE LAST ELEMENT IN NB
        LD      (HL),A          ; SAVE THE NODEID
;
; PREPARE TO ENTER THE COMPARISON ROUTINE
;
        LD      HL,NAADDR       ; POINT TO THE START OF THE NA ARRAY
        JP      COMPAR          ; START COMPARING
;
```

```
;  THIS SECTION IS ENTERED TO CONTINUE A PREVIOUS SEARCH
;
IROWNZ: LD      L,(IY+0)        ; LOAD LOW ORDER BYTE OF IROW INTO L
        LD      H,(IY+1)        ; LOAD HIGH ORDER BYTE OF IROW INTO H
        ADD     HL,HL           ; DOUBLE IROW I.E. HL=IROW*2
;
;  CALCULATE ABSOLUTE ADDRESS OF NEXT NODE TO BE SEARCHED
;
        LD      A,(NBFLAG)      ; GET THE NB FLAG
        CP      00H             ; IS IT SET?
        JP      NZ,NBOFST       ; NO-CALCULATE OFFSET IN NB ARRAY
        LD      DE,NAADDR       ; OTHERWISE CALCULATE OFFSET IN NA ARRAY
NAOFST: ADD     HL,DE           ; FORM IROW + NAADDR
        JP      ENTCOM          ; AND ENTER THE COMPARISON ROUTINE
;
NBOFST: LD      DE,NBADDR       ; GET THE STARTING ADDRESS OF THE NB ARRAY
        JP      NAOFST          ; AND DO SAME AS FOR NAADDR
;
ENTCOM: LD      A,(IX+0)        ; PUT LOW ORDER BYTE OF THE NODE TO BE SEARCHED IN A
;
;  COMPARISON SECTION , HL CONTAINS ABSOLUTE ADDRESS OF NODE TO BE CHECKED
;
COMPAR: CP      A,(HL)          ; COMPARE LOW ORDER BYTE OF ADJOINING NODE WITH A
COMPR1: INC     HL              ; POINT TO
        INC     HL              ; NEXT NODE
        JP      NZ,COMPAR       ; IF NO MATCH DO AGAIN
;
;  AT THIS POINT A MATCH HAS BEEN FOUND -- CHECK IF INTEGER IS LAST NODE
;  I.E. UPPER BYTE =7FH?
;
        DEC     HL              ; POINT TO HIGH ORDER BYTE OF MATCHED NODE
        LD      A,(HL)          ; GET HIGH ORDER BYTE
        CP      A,7FH           ; IS IT =7FH?
        JP      Z,LSTNOD        ; YES - MUST BE THE LAST NODE IN THE ARRAY
        DEC     HL              ; POINT TO LOW ORDER BYTE OF MATCHED NODE
                                ; (SET UP FOR RE-ENTRY INTO COMPARISON ROUTINE)
```

```
                                    ; A STILL CONTAINS HIGH ORDER BYTE
        BIT     7,A                 ; OTHERWISE CHECK IF -VE
                                    ; (I.E. A DOWNSTREAM NODE AND THEFORE AN INVALID NODE
        JP      NZ,COMPR1           ; IF -VE THEN START COMPARISON WITH NEXT NODE
        JP      VALIDN              ; OTHERWISE A VALID NODE NUMBER HAS BEEN FOUND -
                                    ; PROCESS IT AND RETURN TO CALLING PROGRAM
;
; THIS SECTION IS ENTERED WHEN THE SEARCH OF THE NA AND NB ARRAYS HAS BEEN COMPLETED
;
LSTNOD: LD      A,(NBFLAG)          ; GET THE NBFLAG
        CP      A,00H               ; IS IT 0?
        JP      Z,SRCHNB            ; YES - GO TO SET NB FLAG AND START SEARCHING NB ARRAY
;
; AT THIS POINT THE END OF THE NB ARRAY HAS BEEN REACHED THERFORE SET IROW TO ZERO AND RETURN
;
        XOR     A,A                 ; ZERO A
        LD      (IY+0),A            ; AND ZERO OUT LOWER AND UPPER
        LD      (IY+1),A            ; IROW BYTES
        RET                         ; AND RETURN
;
; AT THIS POINT THE END OF THE NA ARRAY HAS BEEN REACHED THEREFORE START SEARCHING NB
;
SRCHNB: LD      A,0FFH              ; SET THE NBFLAG AND
        LD      (NBFLAG),A          ; STORE
        LD      HL,NBADDR           ; POINT TO THE START OF THE NB ARRAY
        LD      A,(IX+0)            ; GET THE NODEID IN A AGAIN
        JP      COMPAR              ; AND START COMPARING
;
; AT THIS POINT A VALID NODE NUMBER HAS BEEN FOUND WHOSE ABSOLUTE ADDRESS IS IN HL
;
VALIDN: LD      A,(NBFLAG)          ; GET THE NB FLAG
        CP      A,0FFH              ; IS IT SET?
        JP      Z,LOADNB            ; YES - CALCULATE RELATIVE ADDRESS IN THE NB ARRAY
;
; OTHERWISE A VALID NODE HAS BEEN FOUND IN THE NA ARRAY
;
```

```
          LD        DE,NAADDR       ; OTHERWISE CALCULATE RELATIVE ADDRESS IN THE NA ARRAY
          LD        BC,NBADDR       ; PREPARE TO FIND ADJOINING NODE IN NB ARRAY
          JP        SUBTRT          ; CALCULATE RELATIVE ADDRESS
;
; VALID NODE HAS BEEN FOUND IN THE NB ARRAY
;
LOADNB:   LD        DE,NBADDR       ; PREPARE TO CALCULATE RELATIVE ADDRESS IN THE NB ARRAY
          LD        BC,NAADDR       ; PREPARE TO FIND ADJOINING NODE IN THE NA ARRAY
SUBTRT:   SCF                       ; CLEAR THE CARRY FLAG SO IT
          CCF                       ; DOESN'T AFFECT THE SUBTRACT OPERATION
          SBC       HL,DE           ; CALCULATE RELATIVE ADDRESS
          LD        D,H             ; COPY THE
          LD        E,L             ; RELATIVE ADDRESS TO DE
          ADD       HL,BC           ; CALCULATE ABSOLUTE ADDRESS OF ADJOINING NODE
          INC       HL              ; POINT TO THE HIGH ORDER BYTE OF NODE
          LD        A,(HL)          ; GET HIGH ORDER BYTE OF ADJOINING NODE
          DEC       HL              ; POINT TO LOW ORDER BYTE AGAIN
          BIT       7,A             ; IS NODE NEGATIVE?
          LD        A,(HL)          ; GET LOW ORDER BYTE IN A AGAIN
          JP        Z,STRNOD        ; NODE NOT NEGATIVE - DON'T  TAKE TWO'S COMPLEMENT
          NEG                       ; OTHERWISE TAKE TWO'S COMPLEMENT
STRNOD:   LD        IY,(NODADR)     ; GET THE NODE PARAMETER ADDRESS
          LD        (IY+0),A        ; AND SAVE ADJOINING NODE NUMBER IN NODE
          XOR       A,A             ; ENSURE UPPER BYTE OF
          LD        (IY+1),A        ; NODE IS ZERO
          EX        DE,HL           ; RETRIEVE THE RELATIVE ADDRESS
;
; DIVIDE THE RELATIVE ADDRESS BY TWO TO FORM THE IROW PARAMETER
;
          INC       HL              ; SET OFFSET (FORTRAN ARRAYS START AT
          INC       HL              ; ELEMENT NUMBER 1 NOT 0
          LD        IY,(ROWADR)     ; POINT TO LOCATION OF IROW PARAMETER
;
          LD        A,H             ; LOAD,
          SRL       A               ; SHIFT AND
          LD        (IY+1),A        ; STORE HIGH ORDER BYTE
```

```
;
        LD      A,L             ; LOAD,
        RR      A               ; SHIFT AND
        LD      (IY+0),A        ; STORE LOW ORDER BYTE
;
        RET
        END
```

APPENDIX III

FLOATING POINT PROCESSOR LIBRARY

This appendix lists, in Z-80 Assembler language, the floating point processor routines that allows arithmetic to be done by the Intel 8232 Floating Point Processor rather than by software. The functions performed are: multiply, divide, subtract, add, natural logarithms, square roots, and cube roots. Note that the 8232 integrated circuit is identical in function to the 9512 integrated circuit mentioned in the listings. This listing is well documented and self explanatory.

```
        TITLE *** FLOATING POINT PROCESSOR LIBRARY ROUTINES ***
        LIST    ON,NOGEN
;
        ENTRY   $MB,$AB,$SB,$DB,ELOG,ESQRT,CBRT
        EXT     $AC
        TITLE *** FPP ADDRESSES AND MACROS ***
;
; 9512 PORT ADDRESSES
;
D9512:  EQU     58H             ; FPP DATA PORT
C9512:  EQU     59H             ; FPP COMMAND PORT
;
;                       MACRO DEFINITIONS
;
SADD:   MACRO                   ; SINGLE PRECISION ADD
        LD      A,01H
        OUT     (C9512),A
        MEND
SSUB:   MACRO                   ; SINGLE PRECISION SUBTRACT
        LD      A,02H
        OUT     (C9512),A
        MEND
SMUL:   MACRO                   ; SINGLE PRECISION MULTIPLY
        LD      A,03H
        OUT     (C9512),A
        MEND
SDIV:   MACRO                   ; SINGLE PRECISION DIVIDE
        LD      A,04H
        OUT     (C9512),A
        MEND
CHSS:   MACRO                   ; SINGLE PRECISION CHANGE SIGN
        LD      A,05H
        OUT     (C9512),A
        MEND
PTOS:   MACRO                   ; SINGLE PRECISION PUSH
        LD      A,06H
```

```
        OUT     (C9512),A
        MEND
POPS:   MACRO                   ; SINGLE PRECISION POP
        LD      A,07H
        OUT     (C9512),A
        MEND
XCHS:   MACRO                   ; SINGLE PRECISION EXCHANGE TOS AND NOS
        LD      A,08H
        OUT     (C9512),A
        MEND
CHSD:   MACRO                   ; DOUBLE PRECISION CHANGE SIGN
        LD      A,2DH
        OUT     (C9512),A
        MEND
PTOD:   MACRO                   ; DOUBLE PRECISION PUSH
        LD      A,2EH
        OUT     (C9512),A
        MEND
POPD:   MACRO                   ; DOUBLE PRECISION POP
        LD      A,2FH
        OUT     (C9512),A
        MEND
CLR:    MACRO                   ; CLEAR STATUS
        LD      A,00H
        OUT     (C9512),A
        MEND
DADD:   MACRO                   ; DOUBLE PRECISION ADD
        LD      A,29H
        OUT     (C9512),A
        MEND
DSUB:   MACRO                   ; DOUBLE PRECISION SUBTRACT
        LD      A,2AH
        OUT     (C9512),A
        MEND
DMUL:   MACRO                   ; DOUBLE PRECISION MULTIPLY
        LD      A,2BH
```

```
        OUT     (C9512),A
        MEND
DDIV:   MACRO                   ; DOUBLE PRECISION DIVIDE
        LD      A,2CH
        OUT     (C9512),A
        MEND
LDFPP:  MACRO
;
; THIS MACRO CONVERTS FORTRAN SINGLE PRECISION NUMBERS TO FPP REPRE-
; SENTATION AN LOADS THE FOUR BYTES INTO THE FPP (9512 TYPE).
; C MUST CONTAIN THE ADDRESS OF THE FPP DATA PORT. HL MUST POINT TO
; THE LSB OF THE NUMBER.
;
        OUTI                    ; MSB-3 TO THE FPP
        OUTI                    ; MSB-2 TO THE FPP
        LD      A,(HL)          ; MSB-1 IN A
        INC     HL              ; POINT TO MSB
        LD      D,(HL)          ; MSB IN D
        INC     D               ; CHECK IF EXPONENT IS ZERO
        DEC     D               ; I.E. ENTIRE NUMBER IS ZERO
        JR      Z,$+04H         ; YES - DON'T ADJUST THE EXPONENT
        DEC     D               ; ADJUST THE
        DEC     D               ; EXPONENT
        RLA                     ; BIT 7 OF MSB-1 MOVED TO CARRY
        RR      D               ; CARRY TO BIT 7 OF MSB, BIT 0 OF MSB TO CARRY
        RRA                     ; CARRY TO BIT 7 OF MSB-1
        OUT     (D9512),A       ; MSB-1 TO THE FPP
        OUT     (C),D           ; MSB TO THE FPP
        MEND
;
UNLFPP: MACRO
;
; THIS MACRO CONVERTS SINGLE PRECISION FPP REPRESENTATION NUMBERS TO
; FORTRAN REPRESENTATION.  TRANSFER OF DATA FROM THE FPP (9512 TYPE) TO
; $AC IS ALSO DONE.  C MUST CONTAIN THE ADDRESS OF THE FPP DATA PORT.
; HL MUST POINT TO THE MSB OF $AC.
```

```
;
        IN      D,(C)           ; MSB TO D
        IN      A,(D9512)       ; MSB-1 TO A
        RLA                     ; BIT 7 OF MSB-1 TO CARRY
        RL      D               ; CARRY TO BIT 0 OF MSB, BIT 7 OF MSB TO CARRY
        RRA                     ; CARRY TO BIT 7 OF MSB-1
        INC     D               ; CHECK IF EXPONENT IS ZERO
        DEC     D               ; I.E. IS ENTIRE NUMBER ZERO
        JR      Z,$+04H         ; YES - DON'T ADJUST EXPONENT
        INC     D               ; ADJUST THE
        INC     D               ; EXPONENT
        LD      (HL),D          ; MSB TO $AC+3
        DEC     HL              ; HL=$AC+2
        LD      (HL),A          ; MSB-1 TO $AC+2
        DEC     HL              ; HL=$AC+1
        IND                     ; MSB-2 TO $AC+1
        IND                     ; MSB-3 TO $AC
        MEND
        TITLE *** $MB - SINGLE PRECISION MULTIPLY ***
;
; MULTIPLY TWO SINGLE PRECISION NUMBERS
;
$MB:    LD      C,D9512
        LDFPP                   ; LOAD THE FIRST OPERAND (ADDRESS IS IN HL)
        LD      HL,$AC          ; POINT TO SECOND OPERAND
        LDFPP                   ; LOAD SECOND OPERAND
        SMUL                    ; AND MULTIPLY
        UNLFPP                  ; UNLOAD RESULT TO $AC
        RET
        TITLE *** $AB - SINGLE PRECISION ADD ***
;
; ADD TWO SINGLE PRECISION NUMBERS
;
$AB:    LD      C,D9512
        LDFPP
        LD      HL,$AC
```

```
        LDFPP
        SADD
        UNLFPP
        RET
        TITLE *** $SB - SINGLE PRECISION SUBTRACT ***
;
; SUBTRACT TWO SINGLE PRECISION NUMBERS
;
$SB:    LD      C,D9512
        LDFPP
        LD      HL,$AC
        LDFPP
        XCHS                    ; SWAP OPERANDS SO OPERATION DONE IN RIGHT ORDER
        SSUB
        UNLFPP
        RET
        TITLE *** $DB - SINGLE PRECISION DIVIDE ***
;
; DIVIDE TWO SINGLE PRECISION NUMBERS
;
$DB:    LD      C,D9512
        LDFPP
        LD      HL,$AC
        LDFPP
        XCHS                    ; SWAP OPERANDS SO OPERATION DONE IN RIGHT ORDER
        SDIV
        UNLFPP
        RET
        TITLE *** ELOG - SINGLE PRECISION NATURAL LOGARITHM ***
;
; THE FOLLOWING CONSTANTS ARE IN FPP FORMAT
;
ONE:    DEFB    00H,00H,80H,3FH         ;1.0
PT25:   DEFB    00H,00H,80H,3EH         ;0.25
LOG2:   DEFB    18H,72H,31H,3FH         ;0.69314718
LOG2D2: DEFB    18H,72H,0B1H,0BEH       ;-0.34657359
```

```
RT2D2:  DEFB    0F2H,04H,35H,3FH        ;1.4142135/2.0
C0:     DEFB    00H,00H,00H,40H         ;1.999999993743
C1:     DEFB    0DAH,0AAH,2AH,3FH       ;0.666669484507
C2:     DEFB    0F7H,9FH,0CCH,3EH       ;0.39965794919
C3:     DEFB    1AH,1DH,9AH,3EH         ;0.301003281
A1:     DEFB    00H,00H,00H,00H         ;0.0 INITIALLY
;
EMSG1:  DEFB    '**FPP-LG**',0DH,0AH,'$'
;
; DEFINE SCRATCHPAD AREAS
;
ADDR:   DS      2
FLAG:   DS      1
W1:     DS      4
W4:     DS      4
Y:      DS      4
;
; MACROS DEFINITIONS
;
RDSTAT: MACRO                           ; GET FPP STATUS IN A
        IN      A,(C9512)
        MEND
;
PUT:    MACRO   #SRC                    ; PUT FPP FORMATTED DATA IN THE FPP
        LD      HL,#SRC                 ; ASSUMES C CONTAINS D9512
        OUTI                            ; B CONTENTS ARE DESTROYED
        OUTI
        OUTI
        OUTI
        MEND
;
GET:    MACRO   #DEST                   ; GET DATA FROM FPP, NO CONVERSION
        LD      HL,#DEST+3              ; ASSUMES C CONTAINS D9512
        IND                             ; B CONTENTS ARE DESTROYED
        IND
        IND
```

```
        IND
        MEND
;
ELOG:   LD      (ADDR),HL       ; SAVE ARGUEMENTS ADDRESS
        LD      A,00H           ; ZERO OUT
        LD      (FLAG),A        ; FLAG
        LD      C,(D9512)       ; PREPARE FOR LDFPP MACRO
        LDFPP                   ; AND LOAD IT X,?,?,?
        PTOS                    ; X,X,?,?
        PUT     ONE             ; 1,X,X,?
        SSUB                    ; X-1,X,?,?
;
; WHILE THE FPP IS SUBTRACTING CHECK IF THE ARGUEMENT IS VALID
;
        LD      HL,(ADDR)       ; POINT TO THE ARGUEMENT
        INC     HL              ; POINT TO MSB
        INC     HL              ; OF THE MANTISSA
        BIT     7,(HL)          ; CHECK THE SIGN BIT
        JP      NZ,ERR1         ; IF -VE THEN ERROR
        INC     HL              ; OTHERWISE CHECK
        LD      A,(HL)          ; IF EXPONENT
        CP      A,00H           ; IS ALL ZEROS, I.E. NUMBER  IS ZERO
        JP      Z,ERR1          ; YES?-ERROR
;
; BACK TO THE FPP SUBTRACTION SHOULD BE FINISHED BY NOW
;
        PTOS                    ; X-1,X-1,X,?
AGAIN:  RDSTAT                  ; READ STATUS REGISTER
        BIT     7,A             ; CHECK IF DONE
        JR      NZ,AGAIN        ; NO?-CHECK AGAIN
        BIT     6,A             ; CHECK IF X-1 IS -VE
        JR      Z,CONT          ; NO-SKIP NEXT INSTRUCTION
        CHSS                    ; OTHERWISE MAKE X-1 +VE I.E. ABS(X-1)
CONT:   PUT     PT25            ; 0.25,ABS(X-1),X-1,X
        SSUB                    ; ABS(X-1)-.25,X-1,X,?
;
```

```
;  WHILE THE FPP IS SUBTRACTING PREPARE THE FRACTIONAL PART OF THE ARGUEMENT
;
        LD      HL,(ADDR)       ; POINT TO SOURCE
        LD      DE,Y            ; AND DESTINATION
        LD      BC,04H          ; NUMBER OF BYTES TO TRANSFER
        LDIR                    ; MOVE ARGUEMENT'S CONTENTS TO Y
        DEC     DE              ; POINT TO EXPONENT
        LD      A,80H           ; AND STORE AN
        LD      (DE),A          ; EXPONENT OF ZERO
        LD      C,D9512         ; POINT TO FPP DATA PORT
;
WAIT:   RDSTAT                  ; IS CURRENT
        BIT     7,A             ; COMMAND FINISHED?
        JR      NZ,WAIT         ; NO?-CHECK AGAIN
        BIT     6,A             ; CHECK IF RESULT IS +VE
        JR      Z,FRMLA2        ; YES?-USE FORMULA 2
;
;  AT THIS POINT ABS(X-1).LE.0.25. THEREFORE CALCULATE W=(X-1)/(X+1)
;  AND LOG(X)=W*SUM(C(I)*W**(2*I))
;
        POPS                    ; X-1,X,?,?
        PTOD                    ; X-1,X,X-1,X
        POPS                    ; X,X-1,X,X-1
        PUT     ONE             ; 1.0,X,X-1,X
        SADD                    ; X+1.0,X-1,X,?
        SDIV                    ; (X-1.0)/(X+1.0)=W,X,?,?
        PTOS                    ; W,W,X,?
        OUT     (C9512),A       ; W,W,W,X
        GET     W1              ; W,W,X,?
        LD      A,0FFH          ; SET FLAG TO INDICATE
        LD      (FLAG),A        ; THAT CONSTANTS MUST NOT BE ADDED TO SERIES
        JP      SERIES          ; GO TO CALCULATE THE SERIES
;
;  THIS SECTION USES THE SECOND FORMULA FOR LOGX
;  CALCULATE W=(Y-RT2D2)/(Y+RT2D2) AND LOG(X)=
;  A*LOG(2)-1/2*LOG(2)+W*SUM(C(I)*W**(2*I))
```

```
;
FRMLA2: LD      HL,Y            ; POINT TO Y
        LD      C,D9512         ; POINT TO FPP DATA PORT
        LDFPP                   ; Y,?,?,?
        PTOS                    ; Y,Y,?,?
        PUT     RT2D2           ; RT2D2,Y,Y,?
        SSUB                    ; Y-RT2D2,Y,?,?
        PTOD                    ; Y-RT2D2,Y,Y-RT2D2,Y
        POPS                    ; Y,Y-RT2D2,Y,Y-RT2D2
        PUT     RT2D2           ; RT2D2,Y,Y-RT2D2,Y
        SADD                    ; Y+RT2D2,Y-RT2D2,Y,?
        SDIV                    ; (Y-RT2D2)/(Y+RT2D2)=W,Y,?,?
;
; THIS SECTION CONVERTS THE ARGUEMENTS EXPONENT INTO A FLOATING POINT
; NUMBER IN FPP FORMAT.  IT SHOULD COMPLETE JUST BEFORE SDIV IS FINISHED.
;
        LD      HL,(ADDR)       ; POINT TO THE ARGUEMENT
        INC     HL              ; AND THEN
        INC     HL              ; POINT TO
        INC     HL              ; THE EXPONENT
        LD      A,(HL)          ; GET THE EXPONENT
        CP      A,80H           ; AND SEE IF IT IS 0
        JR      NZ,LBL1         ; NO?-GO TO HANDLE NON-ZERO EXPONENT
        LD      D,00H           ; ZERO THE EXPONENTS EXPONENT
        LD      E,00H           ; AND THE MANTISSA
        JR      STORE           ; THEN STORE IN A1
;
; ENTRY POINT FOR NON-ZERO EXPONENTS
;
LBL1:   LD      B,00H           ; RESET THE MANTISSA SIGN FLAG
        LD      D,86H           ; SET D THIS WILL BECOME THE EXP. EXP.
        LD      E,(HL)          ; GET THE EXPONENT
        BIT     7,E             ; CHECK EXP SIGN 1=+VE, 0=-VE
        JR      NZ,LBL2         ; IF +VE THEN SKIP THE NEGATE
;
; AT THIS POINT BIT 7 OF E=0 I.E. EXPONENT IS NEGATIVE.
```

```
;  THEREFORE SET BIT 7 OF D TO MAKE MANTISSA OF A1 NEGATIVE AND MAKE
;  E +VE BY TAKING TWO'S COMPLEMENT
;
        LD      B,0FFH          ; SET MANTISSA SIGN FLAG
        LD      A,E             ; AND THEN
        NEG                     ; TAKE TWO'S COMPLEMENT
        LD      E,A             ; OF THE EXPONENT
LBL2:   SLA     E               ; DISCARD THE EXPONENTS SIGN BIT
LBL3:   DEC     D               ; ADJUST EXPONENT
        SLA     E               ; SHIFT LEFT, 0 FILL
        JR      NC,LBL3         ; UNTIL CARRY=1
;
;  AT THIS POINT A1 HAS BEEN FORMED NORMALIZED TO 0.5 .LE. MANTISSA.LT.1.0
;  MUST STILL PUT B0 OF EXPONENT IN B7 OF MSB OF MANTISSA TO MATCH FPP
;  REPRESENTATION.
;
        SRL     D               ; BIT 0 OF D TO CARRY, 0 TO B7
        RR      E               ; CARRY TO BIT 7 OF E
        BIT     7,B             ; CHECK IF MANTISSA SIGN FLAG SET
        JR      Z,STORE         ; NO?-THEN LEAVE MANTISSA SIGN AS A 0 (+VE)
        SET     7,D             ; OTHERWISE SET SIGN 1=-VE
;
;  NOW STORE THE RESULTS IN A1, NOTE THAT MSB-3 AND MSB-2 ARE ALWAYS 0
;
STORE:  LD      HL,A1+2         ; POINT TO MANTISSA OF A1
        LD      (HL),E          ; AND STORE MANTISSA
        INC     HL              ; POINT TO EXPONENT
        LD      (HL),D          ; AND STORE EXPONENT
;
;  FINISHED ADJUSTING EXPONENT
;
        PTOS                    ; W,W,Y,?
        PTOS                    ; W,W,W,Y
        GET     W1              ; W,W,Y,?
;
;  CALCULATE THE SERIES W*SUM(C(I)*W**(2*I))
```

```
;
SERIES: SMUL                    ; W*W,?,?,?
        PTOS                    ; W**2,W**2,?,?
        PUT     C1              ; C1,W**2,W**2,?
        SMUL                    ; C1*W**2,W**2,?,?
        PUT     C0              ; C0,C1*W**2,W**2,?
        SADD                    ; C0+C1*W**2,W**2,?,?
        PTOD                    ; SUM,W**2,SUM,W**2
        POPS                    ; W**2,SUM,W**2,SUM
        PTOS                    ; W**2,W**2,SUM,W**2
        SMUL                    ; W**4,SUM,W**2,?
        PTOS                    ; W**4,W**4,SUM,W**2
        GET     W4              ; W**4,SUM,W**2,?
        PUT     C2              ; C2,W**4,SUM,W**2
        SMUL                    ; C2*W**4,SUM,W**2,?
        SADD                    ; SUM,W**2,?,?
        PTOD                    ; SUM,W**2,SUM,W**2
        POPS                    ; W**2,SUM,W**2,SUM
        PUT     W4              ; W**4,W**2,SUM,W**2
        SMUL                    ; W**6,SUM,W**2,?
;
; CHECK FOR AN UNDERFLOW. THIS OCCURS WHEN W IS VERY SMALL
; I.E. Y ALMOST EQUALS RT2D2,THEREFORE C3*W**6 WILL CAUSE AN UNDERFLOW
;
WAIT2:  RDSTAT                  ; WAIT FOR CURRENT
        BIT     7,A             ; COMMAND TO
        JR      NZ,WAIT2        ; COMPLETE
        BIT     2,A             ; CHECK FOR UNDERFLOW
        JR      Z,CONT2         ; OK? GO TO MULTIPLY WITH C3
        POPS                    ; SUM,W**2,?,?
        JR      CONT4           ; IGNORE C3
CONT2:  PUT     C3              ; C3,W**6,SUM,?
        SMUL                    ; C3*W**6,SUM,?,?
WAIT3:  RDSTAT                  ; WAIT FOR CURRENT
        BIT     7,A             ; COMMAND
        JR      NZ,WAIT3        ; TO COMPLETE
```

```
          BIT     2,A             ; CHECK FOR UNDERFLOW
          JR      Z,CONT3         ; OK? GO TO SUM
          POPS                    ; SUM,?,?,?
          JR      Z,CONT4         ;
CONT3:    SADD                    ; SUM,?,?,?
CONT4:    PUT     W1              ; W,SUM,?,?
          SMUL                    ; SUM,W**2,?,?
;
          LD      A,(FLAG)        ; CHECK IF OTHER CONSTANTS
          CP      A,0FFH          ; MUST BE ADDED IN
          JP      Z,WRAPUP        ; NO?-GO TO FINISH
          PUT     LOG2D2          ; LOG2D2,SUM,W**2,?
          SADD                    ; LOG2D2+SUM,W**2,?,?
          PUT     LOG2            ; LOG2,SUM,W**2,?
          PUT     A1              ; A1,LOG2,SUM,W**2
          SMUL                    ; A1*LOG2,SUM,W**2,?
          SADD                    ; A1*LOG2+SUM,W**2,?,?
;
WRAPUP:   LD      HL,$AC          ; FORTRAN FUNCTIONS MUST RETURN A REAL VALUE IN $AC
          INC     HL
          INC     HL
          INC     HL
          UNLFPP
          RET
; ************************************************************
ERR1:     LD      C,09H           ; SYSTEM CALL
          LD      DE,EMSG1        ; TO PRINT **LG**
          CALL    05H             ; PRINT
          LD      C,00H           ; AND
          CALL    05H             ; ABORT PROGRAM
          TITLE *** ESQRT - SINGLE PRECISION SQUARE ROOT ***
;
; ***************** SQRT ***********************************************
;
; THE FOLLOWING CONSTANTS ARE IN FPP FORMAT
;
```

```
PT4173: DEFB    0A8H,0A9H,0D5H,3EH      ; 0.41731
PT5901: DEFB    0B9H,14H,17H,3FH        ; 0.59016
RTPT5:  DEFB    0F3H,04H,35H,3FH        ; 0.7071067811
RF:     DS      4
RY0:    DS      4
XP0:    DEFB    58H             ; WILL BE LOADED INTO C LATER
XP1:    DEFB    00H             ; STORAGE FOR THE EXPONENT
;
; ERROR MESSAGE
;
RTERR:  DEFB    '**FPP-SQRT**',0DH,0AH,'$'
;
; TO SAVE TIME MODIFIED VERSIONS OF LDFPP AND UNLFPP ARE USED.  NOTE
; THAT HL CONTAINS THE ADDRESS OF THE MSB-1 OF THE ARGUEMENT.
;
; FIRST LOAD THE FRACTIONAL PART OF THE ARGUEMENT, MODIFY THE EXPONENT
; SO THE FPP GETS A FRACTION.
;
ESQRT:  LD      C,(D9512)       ; INITIALIZE THE FPP DATA PORT ADDRESS
        OUTI                    ; MSB-3 TO THE FPP
        OUTI                    ; MSB-2 TO THE FPP
        LD      A,(HL)          ; MSB-1 IN A
        PUSH    HL              ; SAVE HL, CURRENTLY POINTING TO MSB-1
;
; THE EXPONENT MUST BE SET TO GIVE A FRACTIONAL VALUE ONLY.  I.E. 7FH=0
; IN FPP EXPONENT NOTATION, BUT THE FORTRAN VERSION HAS BEEN LOADED AS
; 0.5.LE.X.LT.1.0.  THE FPP WANTS IT'S NUMBERS AS 1.0.LE.X.LT.2.0
; THEREFORE THE CORRECT FPP EXPONENT SHOULD BE 7EH.
;
        LD      D,7EH           ; FPP EXPONENT OF -1
        RLA                     ; B7 OF MSB-1 TO CY
        RR      D               ; CY-B7 OF MSB, B0 OF MSB-CY
        RRA                     ; CY-B7 OF MSB-1
        OUT     (D9512),A       ; MSB-1 TO FPP
        OUT     (C),D           ; MSB TO FPP
;
```

```
; FORM 0.59016*F
;
        PTOS                    ; F,F,?,?
        PUT     PT5901          ; 0.59016,F,F,?
        SMUL                    ; 0.59016*F,F,?,?
;
; WHILE FPP IS MULTIPLYING CHECK IF ARGUEMENT IS VALID.  NOTE THAT HL
; IS POINTING TO MSB-1.  IF X.LT.0.0 THEN ERROR, IF X.EQ.0.0 SET RESULT
; TO 0.0 AND RETURN
;
        POP     HL              ; RESTORE POINTER TO MSB-1
        BIT     7,(HL)          ; MANTISSA -VE? (0=+VE,1=-VE)
        JP      NZ,SQERR        ; IF -VE THEN ERROR
        INC     HL              ; POINT TO THE EXPONENT
        LD      A,(HL)          ; AND LOAD IN A
        CP      A,00H           ; IS IT 0
        JR      NZ,CONT10       ; NO?-CONTINUE WITH CALCULATIONS
        LD      HL,$AC          ; OTHERWISE ZERO THE RESULT
        INC     HL              ; POINT
        INC     HL              ; TO
        INC     HL              ; THE EXPONENT
        LD      (HL),A          ; SET NUMBER TO ZERO
        RET                     ; AND RETURN
;
CONT10: PUSH    HL              ; SAVE EXPONENTS ADDRESS
        PUT     PT4173          ; 0.41731,0.59016*F,F,?
        SADD                    ; Y0,F,?,?
        GET     RY0             ; F,?,?,Y0
        GET     RF              ; ?,?,Y0,F
        POPS                    ; ?,Y0,F,?
        POPS                    ; Y0,F,?,?
        PUT     RF              ; F,Y0,F,?
        PUT     RY0             ; Y0,F,Y0,F
        SDIV                    ; F/Y0,Y0,F,?
        POP     HL              ; RESTORE POINTER TO THE EXPONENT
;
```

```
;  WHILE THE FPP IS DIVIDING SET UP THE EXPONENT.  NOTE THAT
;  HL IS POINTING TO THE EXPONENT
;
;  SET UP SO 1/(SQRT(2)) IS NOT USED I.E ASSUME EXP IS EVEN
;  SET UP TO ADD A,B I.E. ASSUME EXP IS POSITIVE
;
        LD      DE,0F18H        ; OPCODE FOR JR $+0FH (WRITTEN BACKWARDS)
        LD      (LOC25),DE      ; AND STORE SO RTPT5 IS NOT USED
        LD      A,80H           ; OPCODE FOR ADD A,B
        LD      (ADDSUB),A      ; AND STORE
        LD      A,(HL)          ; GET THE EXPONENT
;
        BIT     0,A             ; IS EXP EVEN?
        JR      Z,LBL38         ; YES?-SKIP
        INC     A               ; OTHERWISE FORM EXP+1
        LD      DE,0000H        ; OPCODE FOR NOP,NOP
        LD      (LOC25),DE      ; AND STORE SO RTPT5 IS USED
;
LBL38:  BIT     7,A             ; IS EXP +VE
        JR      NZ,LBL39        ; YES?-SKIP
        NEG                     ; CONVERT TO A +VE EXPONENT
        PUSH    AF              ; SAVE A
        LD      A,90H           ; OPCODE FOR SUB A,B
        LD      (ADDSUB),A      ; AND STORE
        POP     AF              ; RESTORE A
LBL39:  RES     7,A             ; DELETE SIGN BIT
        SRL     A               ; FORM EXP/2 OR (EXP+1)/2
        LD      (XP1),A         ; SAVE THE EXP FOR LATER USE
;
        SADD                    ; Z,F,?,?
        PTOS                    ; Z,Z,F,?
;
;  FORM Z*0.25
;
;  NOTE THAT MSB'S BO IS ACTUALLY IN MSB-1 INSIDE THE FPP, I.E. THE
;  EXPONENTS LEAST SIGNIFICANT BIT IS IN THE MOST SIGNIFICANT BIT OF
```

```
; MSB-1.
;
        LD      C,(D9512)       ; ENSURE C HAS CORRECT PORT ADDRESS
        IN      A,(C)           ; NOTE THAT YOU CANNOT PARTIALLY UNLOAD
        IN      B,(C)           ; THEN LOAD AGAIN EVEN IF NUMBER OF BYTES
        IN      D,(C)           ; LOADED AND UNLOADED
        IN      E,(C)           ; ARE THE SAME
        OUT     (C),E           ; SOMETHING IN THE FPP
        OUT     (C),D           ; GETS SCREWED UP IF ONLY 1 BYTE IS
        OUT     (C),B           ; LOADED AND THEN LOADED AGAIN
        DEC     A               ; FORM Z*0.25 BY DECREMENTING BIT 0 OF A
        OUT     (C),A           ; EXP TO FPP
;
        POPS                    ; Z,F,?,Z*0.25
        SDIV                    ; F/Z,?,Z*0.25,?
        XCHS                    ; ?,F/Z,Z*0.25,?
        POPS                    ; F/Z,Z*0.25,?,?
        SADD                    ; Y2,?,?,?
;
; LOC25 IS LOADED WITH EITHER A NOP,NOP OR JR $+11H WHILE THE FPP WAS DIVIDING
;
LOC25:  DS      2               ; NOP,NOP OR JR $+11H
;
        PUT     RTPT5           ; 0.7071067,Y2,?,?
        SMUL                    ; RESULT,?,?,?
;
; UNLOAD THE FPP AND AUGMENT THE EXPONENT
;
CONT24: IN      D,(C)           ; MSB TO D
        IN      A,(D9512)       ; MSB-1 TO A
        RLA                     ; B7 OF MSB-1 TO CY
        RL      D               ; CY-B7 OF MSB, B7 OF MSB TO CY
        RRA                     ; CY-B7 OF MAB-1
;
; DON'T BOTHER CHECKING IF NUMBER IS ZERO (IT SHOULD NEVER BE ZERO)
;
```

```
        INC     D               ; ADJUST THE
        INC     D               ; EXPONENT
        LD      E,A             ; SAVE MSB-1
        LD      A,D             ; AND GET MSB (EXPONENT) IN A
        LD      BC,(XPO)        ; GET THE ABSOLUTE VALUE OF THE EXPONENT
;
; ADDSUB WAS LOADED WITH EITHER AN ADD A,B OR A SUB A,B WHILE THE FPP WAS DIVIDING
;
ADDSUB: DS      1               ; ADD A,B OR SUB A,B
;
        LD      HL,$AC          ; POINT
        INC     HL              ; TO THE
        INC     HL              ; ANSWER STORAGE
        INC     HL              ; LOCATION
        LD      (HL),A          ; STORE EXPONENT IN $AC+3
        DEC     HL              ; AND
        LD      (HL),E          ; THE MSB-1 IN $AC+2
        DEC     HL              ; POINT TO $AC+1
        IND                     ; MSB-2 TO $AC+1
        IND                     ; MSB-3 TO $AC
        RET                     ; ALL DONE
;
SQERR:  LD      C,09H           ; SYSTEM CALL
        LD      DE,RTERR        ; TO PRINT ERROR MESSAGE
        CALL    05H             ; PRINT
        LD      C,00H           ; AND
        CALL    05H             ; ABORT PROGRAM
        TITLE *** CBRT - SINGLE PRECISION CUBE ROOT ***
;
; ***************** CBRT ********************************************
;
; THE FOLLOWING CONSTANTS ARE IN FPP FORMAT
;
PT5874: DEFB    0EBH,5FH,16H,3FH        ; 0.5874011
PT4152: DEFB    0EH,0A2H,0D4H,3EH       ; 0.4152989
PT3333: DEFB    0AAH,0AAH,0AAH,3EH      ; 0.3333333
```

```
CBRT21: DEFB    0F6H,2FH,4BH,3FH        ; 0.7937006 = 2**(-1/3)
CBRT22: DEFB    17H,45H,21H,3FH         ; 0.6299605 = 2**(-2/3)
CBRT23: DEFB    17H,45H,0A1H,3FH        ; 1.259921 = 2**(1/3)
CBRT24: DEFB    0F4H,2FH,0CBH,3FH       ; 1.587401 = 2**(2/3)
CF:     DS      4
CY0:    DS      4
CXP0:   DEFB    58H             ; WILL BE LOADED INTO C LATER
CXP1:   DEFB    00H             ; STORAGE FOR THE BINARY VERSION OF THE EXPONENT
FXP:    DS      1               ; FORTRAN VERSION OF THE EXPONENT
DIVFLG: DS      1               ; 0=NO MULTIPLY
                                ; 1=MULTIPLY Y2 BY 2**(-2/3)
                                ; 2=MULTIPLY Y2 BY 2**(-1/3)
TBLE3:  DEFB    0,65,129,       1,66,130,       2,67,131        ; TABLE FOR ABS(EXP)/3
        DEFB    3,68,132,       4,69,133,       5,70,134
        DEFB    6,71,135,       7,72,136,       8,73,137
        DEFB    9,74,138,       10,75,139,      11,76,140
        DEFB    12,77,141,      13,78,142,      14,79,143
        DEFB    15,80,144,      16,81,145,      17,82,146
        DEFB    18,83,147,      19,84,148,      20,85,149
        DEFB    21,86,150,      22,87,151,      23,88,152
        DEFB    24,89,153,      25,90,154,      26,91,155
        DEFB    27,92,156,      28,93,157,      29,94,158
        DEFB    30,95,159,      31,96,160,      32,97,161
        DEFB    33,98,162,      34,99,163,      35,100,164
        DEFB    36,101,165,     37,102,166,     38,103,167
        DEFB    39,104,168,     40,105,169,     41,106,170
        DEFB    42,107,171
;
; ERROR MESSAGE
;
C3ERR:  DEFB    '**FPP-CBRT**',0DH,0AH,'$'
;
; TO SAVE TIME MODIFIED VERSIONS OF LDFPP AND UNLFPP ARE USED.  NOTE
; THAT HL CONTAINS THE ADDRESS OF THE MSB-1 OF THE ARGUEMENT.
;
; FIRST LOAD THE FRACTIONAL PART OF THE ARGUEMENT, MODIFY THE EXPONENT
```

```
; SO THE FPP GETS A FRACTION.
;
CBRT:   LD      C,(D9512)       ; INITIALIZE THE FPP DATA PORT ADDRESS
        OUTI                    ; MSB-3 TO THE FPP
        OUTI                    ; MSB-2 TO THE FPP
        LD      A,(HL)          ; MSB-1 IN A
        PUSH    HL              ; SAVE HL, CURRENTLY POINTING TO MSB-1
;
; THE EXPONENT MUST BE SET TO GIVE A FRACTIONAL VALUE ONLY.  I.E. 7FH=0
; IN FPP EXPONENT NOTATION, BUT THE FORTRAN VERSION HAS BEEN LOADED AS
; 0.5.LE.X.LT.1.0.  THE FPP WANTS IT'S NUMBERS AS 1.0.LE.X.LT.2.0
; THEREFORE THE CORRECT FPP EXPONENT SHOULD BE 7EH.
;
        LD      D,7EH           ; FPP EXPONENT OF -1
        RLA                     ; B7 OF MSB-1 TO CY
        RR      D               ; CY-B7 OF MSB, B0 OF MSB-CY
        RRA                     ; CY-B7 OF MSB-1
        OUT     (D9512),A       ; MSB-1 TO FPP
        OUT     (C),D           ; MSB TO FPP
;
; FORM 0.4152989*F
;
        PTOS                    ; F,F,?,?
        GET     CF              ; GET A COPY OF THE FRACTION
        PTOS                    ; F,F,?,?
        PUT     PT4152          ; 0.4152989,F,F,?
        SMUL                    ; 0.4152989*F,F,?,?
;
; WHILE FPP IS MULTIPLYING CHECK IF ARGUEMENT IS VALID.  NOTE THAT HL
; IS POINTING TO MSB-1.  IF X.LT.0.0 THEN ERROR, IF X.EQ.0.0 SET RESULT
; TO 0.0 AND RETURN
;
        POP     HL              ; RESTORE POINTER TO MSB-1
        BIT     7,(HL)          ; MANTISSA -VE? (0=+VE,1=-VE)
        JP      NZ,CBERR        ; IF -VE THEN ERROR
        INC     HL              ; POINT TO THE EXPONENT
```

```
        LD      A,(HL)          ; AND LOAD IN A
        CP      A,00H           ; IS IT 0
        JR      NZ,CBCT10       ; NO?-CONTINUE WITH CALCULATIONS
        LD      HL,$AC          ; OTHERWISE ZERO THE RESULT
        INC     HL              ; POINT
        INC     HL              ; TO
        INC     HL              ; THE EXPONENT
        LD      (HL),A          ; SET NUMBER TO ZERO
        RET                     ; AND RETURN
;
CBCT10: LD      (FXP),A         ; SAVE FORTRAN VERSION OF THE EXPONENT
        PUT     PT5874          ; 0.5874011,0.4152989*F,F,?
        SADD                    ; Y0,F,?,?
;
        PTOS                    ; Y0,Y0,F,?
        PTOS                    ; Y0,Y0,Y0,F
        POPS                    ; Y0,Y0,F,Y0
        SMUL                    ; Y0*Y0,F,Y0,?
;
; CONVERT THE EXPONENT FROM TWO'S COMPLEMENT TO ABSOLUTE BINARY
;
        LD      A,(FXP)         ; GET THE EXPONENT
        BIT     7,A             ; IS THE EXPONENT .GE. 80H
        JR      NZ,LBL40        ; YES?-GO TO DELETE THE SIGN BIT
;
; FOR NEGATIVE EXPONENTS
;
        NEG                     ; CONVERT TWO'S COMPLEMENT TO ABSOLTUE BINARY
        RES     7,A             ; STRIP THE SIGN BIT
        LD      (CXP1),A        ; SAVE THE EXPONENT
        LD      A,90H           ; OPCODE FOR SUB A,B
        LD      (ADSB),A        ; AND STORE
        JR      CNT200          ; THEN CONTINUE
;
; FOR POSITIVE EXPONENTS
;
```

```
LBL40:  RES     7,A             ; STRIP THE SIGN BIT
        LD      (CXP1),A        ; SAVE THE EXPONENT
        LD      A,80H           ; OPCODE FOR ADD A,B
        LD      (ADSB),A        ; AND STORE
CNT200: SDIV                    ; F/(Y0*Y0),Y0,?,?
;
; DO A TABLE LOOKUP TO FIND IF EXP IS EVENLY DIVISIBLE BY 3
; THIS METHOD, ALTHOUGH IT REQUIRES A LARGE TABLE (128 BYTES)
; IS MUCH FASTER THAN A SOFTWARE DIVIDE. NOTE THAT THE RANGE OF
; DIVIDENDS IS .LT. 127 AND THE DIVISOR IS ALWAYS 3.
;
        LD      HL,0000H        ; ZERO HL
        LD      A,(CXP1)        ; GET THE ABSOLUTE BINARY EXPONENT
        LD      L,A             ; AND STORE IN L
        LD      DE,TBLE3        ; POINT TO THE START OF THE TABLE
        ADD     HL,DE           ; POINT TO THE ANSWER
        LD      A,(HL)          ; GET THE ANSWER
        AND     A,11000000B     ; SAVE BITS 6 AND 7 ONLY
        LD      (DIVFLG),A      ; AND STORE
        LD      A,(HL)          ; RETRIEVE EXP/3
        AND     A,00111111B     ; SAVE B0-B5 ONLY
        LD      (CXP1),A        ; STORE ABSOLUTE BINARY EXPONENT
        XCHS                    ; Y0,F/(Y0*Y0),?,?
;
; MULTIPLY BY 2.0. AFTER UNLOADING STACK STATUS IS F/Y(Y0*Y0),?,?,Y0
; NOTE THAT A COMPLETE 4 BYTE LOAD/UNLOAD MUST BE DONE
;
        IN      D,(C)           ; MSB TO D
        IN      A,(C)           ; MSB-1 TO A
        IN      B,(C)           ; MSB-2 TO B
        IN      E,(C)           ; MSB-3 TO E
        RLA                     ; B7 OF A-CY
        RL      D               ; CY-B0 OF D,B7 OF D-CY
        INC     D               ; MULTIPLY NUMBER BY 2 (CY NOT AFFECTED
                                ; BECAUSE RANGE OF NUMBERS IS .LT.FFH)
        RR      D               ; CY-B7 OF D, B0 OF D TO CY
```

```
        RRA                             ; CY-B7 OF A
        OUT     (C),E                   ; E TO MSB-3
        OUT     (C),B                   ; B TO MSB-2
        OUT     (C),A                   ; A TO MSB-1
        OUT     (C),D                   ; D TO MSB
;
; CURRENT STACK STATUS                  2.0*Y0,F/(Y0*Y0),?,?
;
        SADD                            ; F/(Y0*Y0)+2.0*Y0,?,?,?
        PUT     PT3333                  ; 0.3333,F/(Y0*Y0)+2.0*Y0,?,?
        SMUL                            ; Y1,?,?,?
;
; FORM Y2
;
        PTOS                            ; Y1,Y1,?,?
        PTOS                            ; Y1,Y1,Y1,?
        SMUL                            ; Y1*Y1,Y1,?
        PUT     CF                      ; F,Y1*Y1,Y1,?
        XCHS                            ; Y1*Y1,F,Y1,?
        SDIV                            ; F/(Y1*Y1),Y1,?,?
        XCHS                            ; Y1,F/(Y1*Y1),?,?
;
; MULTIPLY Y1 BY 2.0
;
        IN      D,(C)                   ; MSB TO D
        IN      A,(C)                   ; MSB-1 TO A
        IN      B,(C)                   ; MSB-2 TO B
        IN      E,(C)                   ; MSB-3 TO E
        RLA                             ; B7 OF A-CY
        RL      D                       ; CY-B0 OF D, B7 OF D-CY
        INC     D                       ; MULTIPLY BY 2.0 (CARRY NOT AFFECTED
                                        ; BECAUSE D ALWAYS .LT. FFH)
        RR      D                       ; CY-B7 OF D,B0 OF D TO CY
        RRA                             ; CY-B7 OF A
        OUT     (C),E                   ; E TO MSB-3
        OUT     (C),B                   ; B TO MSB-2
```

```
        OUT     (C),A           ; A TO MSB-1
        OUT     (C),D           ; D TO MSB
;
; CURRENT STACK STATUS          2.0*Y1,F/(Y1*Y1),?,?
;
        SADD                    ; F/(Y1*Y1)+2.0*Y1,?,?
        PUT     PT3333          ; 0.3333,F/(Y1*Y1)+2.0*Y1,?,?
        SMUL                    ; Y2
;
; FOR FORTRAN NEGATIVE EXPONENTS MULTIPLY Y2 BY 2**(2/3) OR 2**(1/3)
; FOR FORTRAN POSITIVE EXPONENTS .GT. 0 MULTIPLY Y2 BY 2**(-2/3) OR 2**(-1/3)
;
        LD      A,(FXP)         ; GET THE FORTRAN EXPONENT
        BIT     7,A             ; CHECK IF SIGN BIT =1 I.E. A ZERO OR +VE EXPOENENT
        JR      NZ,LBL55        ; YES?-GO TO HANDLE 0 OR +VE EXPONENT
;
; CHECK IF MULITPLICATION BY 2**(2/3) OR 2**(1/3) MUST BE DONE
;
        LD      A,(DIVFLG)      ; GET THE DIVIDE FLAG
        CP      A,00H           ; IS IT 0? I.E. NO MULTIPLY REQUIRED
        JR      Z,CONT38        ; YES?-DON'T MULTIPLY
        CP      A,64            ; IS IT 64?
        JR      Z,LBL58         ; YES? GO TO MULTIPLY BY 2**(2/3)
        PUT     CBRT23          ; CBRT23,Y2,?,?
        JR      MLTPLY          ; GO TO MULTIPLY
LBL58:  PUT     CBRT24          ; CBRT24,Y2,?,?
        JR      MLTPLY          ; GO TO MULTIPLY
;
; CHECK IF MULTIPLICATION BY 2**-2/3 OR 2**-1/3 MUST BE DONE
;
LBL55:  LD      A,(DIVFLG)      ; GET THE DIVIDE FLAG
        CP      A,00H           ; ZERO?
        JR      Z,CONT38        ; YES?-DON'T MULTIPLY
        CP      A,64            ; 64?
        JR      Z,LBL48         ; YES?-MULTIPLY BY 2**(-2/3)
        PUT     CBRT21          ; CBRT21,Y2,?,?
```

```
        JR      MLTPLY          ; GO TO MULTIPLY
LBL48:  PUT     CBRT22          ; CBRT22,Y2,?,?
MLTPLY: SMUL                    ; RESULT,?,?,?
;
; UNLOAD THE FPP AND AUGMENT THE EXPONENT
;
CONT38: IN      D,(C)           ; MSB TO D
        IN      A,(D9512)       ; MSB-1 TO A
        RLA                     ; B7 OF MSB-1 TO CY
        RL      D               ; CY-B7 OF MSB, B7 OF MSB TO CY
        RRA                     ; CY-B7 OF MSB-1
;
; DON'T BOTHER CHECKING IF NUMBER IS ZERO (IT SHOULD NEVER BE ZERO)
;
        INC     D               ; ADJUST THE
        INC     D               ; EXPONENT
        LD      E,A             ; SAVE MSB-1
        LD      A,D             ; AND GET MSB (EXPONENT) IN A
        LD      BC,(CXPO)       ; GET THE ABSOLUTE VALUE OF THE EXPONENT
;
; ADDSUB WAS LOADED WITH EITHER AN ADD A,B OR A SUB A,B WHILE THE FPP WAS DIVIDING
;
ADSB:   DS      1               ; ADD A,B OR SUB A,B
;
        LD      HL,$AC          ; POINT
        INC     HL              ; TO THE
        INC     HL              ; ANSWER STORAGE
        INC     HL              ; LOCATION
        LD      (HL),A          ; STORE EXPONENT IN $AC+3
        DEC     HL              ; AND
        LD      (HL),E          ; THE MSB-1 IN $AC+2
        DEC     HL              ; POINT TO $AC+1
        IND                     ; MSB-2 TO $AC+1
        IND                     ; MSB-3 TO $AC
        RET                     ; ALL DONE
;
```

```
CBERR:  LD      C,09H           ; SYSTEM CALL
        LD      DE,C3ERR        ; TO PRINT ERROR MESSAGE
        CALL    05H             ; PRINT
        LD      C,00H           ; AND
        CALL    05H             ; ABORT PROGRAM
        END
```

APPENDIX IV

MANUFACTURER'S DATA ON THE PPG STANDARD ALUMINUM COLLECTOR

This Appendix contains the manufacturer's data on the solar collector modelled in this thesis. This data was used to produce the data required to thermally model the collector.

PPG STANDARD ALUMINUM SOLAR COLLECTORS

The standard aluminum collector is a flat plate-type collector, constructed with the following materials:

1. Unit Size - 34 3/16" x 76 3/16" x 1 5/16"
2. Cover Plates - Two pieces of 1/8-inch clear glass, (Herculite tempered outer lite, Heat Strengthened inner lite)
3. Spacers - Metal tube spacers with desiccant
4. Absorbing Surface - PPG's Duracron Super 600 L/G (UC 40437) flat black coating, a - .95, e = .88
5. Collector Plate - 0.060" Roll-Bond Type 1100 Aluminum absorber plate. Inlet and outlet tubes are 1/2" O.D. aluminum
6. Insulation - 2 1/2" backing of 3# density, fiber glass (K = .24) encased in a 24 gauge painted, galvanized steel pan
7. Edge Framing System - Collectors are available with optional preglazed sash
8. Fluid Capacity - .285 gallons
9. Weight - Approximately 6 lbs./sq. ft. with fluid
10. Pressure Capabilities of Fluid Passages -

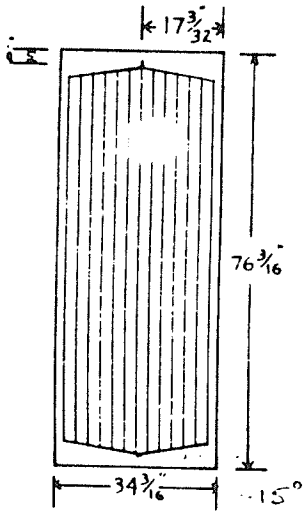
Burst:	500 psi
Bulge:	120 psi
* Working:	100 psi

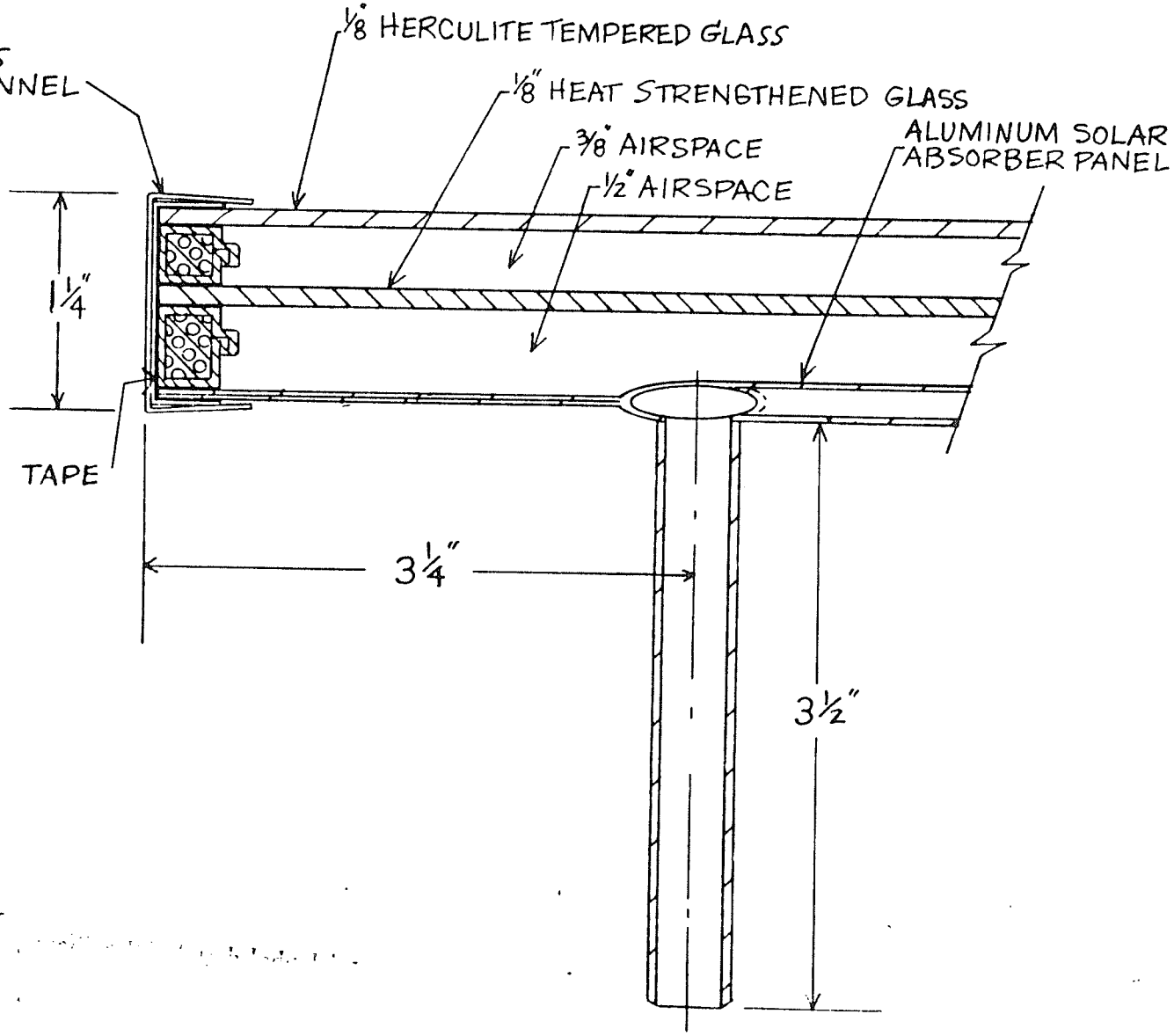
* Calculated, Not Tested

NOTE: Materials and construction subject to change, as new technology directs.

January-1976

PPG ALUMINUM SOLAR COLLECTOR
(FULL SIZE)

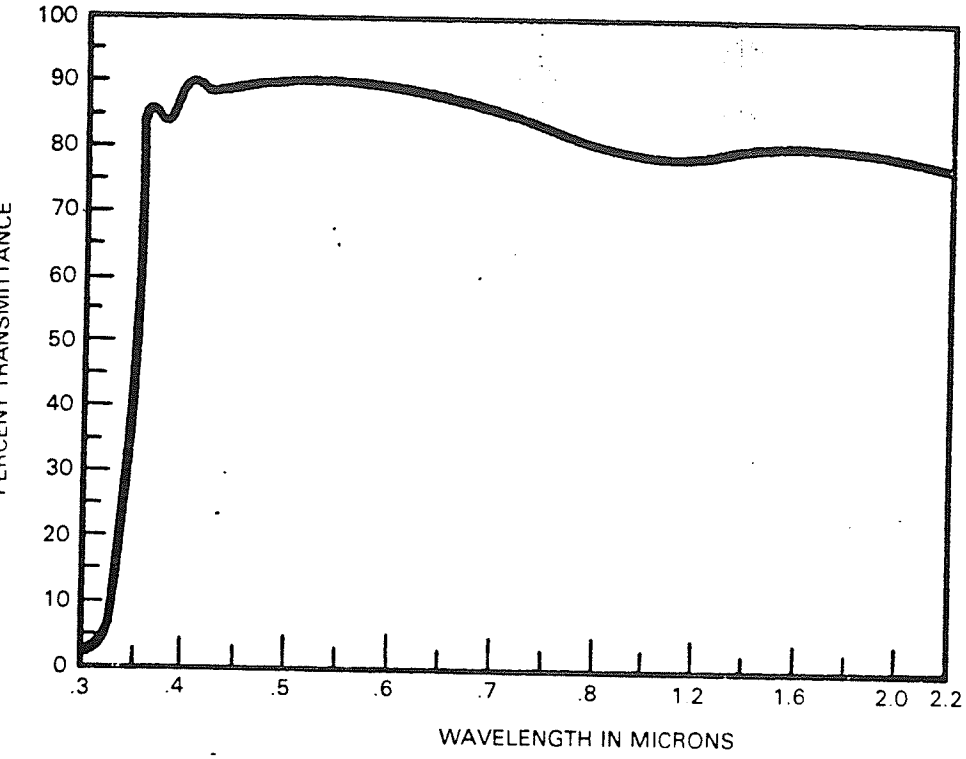




AIV-3

SOLAR SYSTEMS SALES
PPG INDUSTRIES
ONE GATEWAY CENTER
PITTSBURGH, PA. 15222 11/75

igure 7 — Solar Spectral Transmittance of ⅛" Single Light
lear HERCULITE Tempered Glass



PROPERTIES OF PPG BASELINE SOLAR COLLECTOR

1. Glass Cover Plates: PPG's Baseline solar collector utilizes ⅛ inch HERCULITE® fully tempered clear float glass because it provides high solar transmittance as shown in the figure 7 and is opaque in the reradiating infrared spectrum i.e. beyond 2.8 microns.

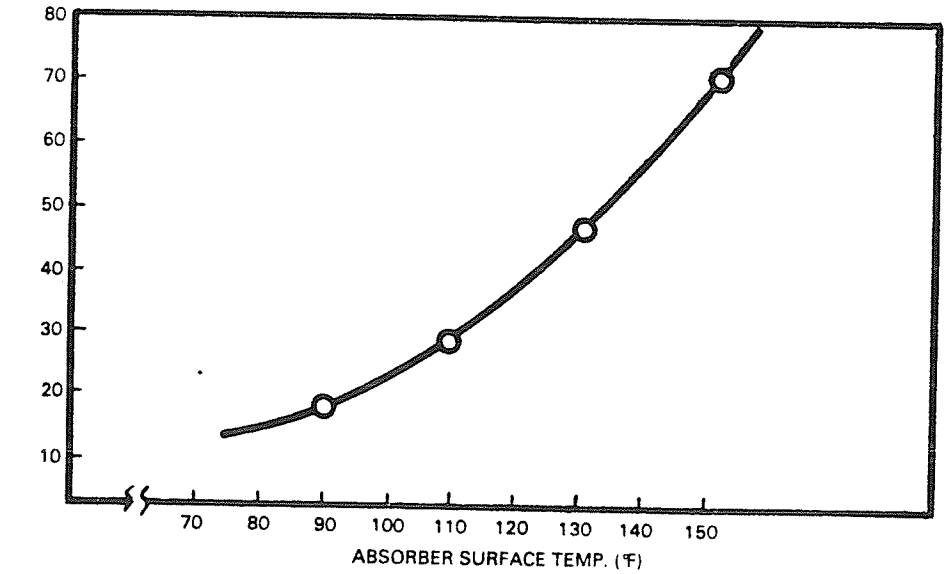
2. Thermal Properties:

a) Thermal losses Two cover plates are provided to reduce upward heat losses. Figure 8 shows typical heat loss curve for PPG Baseline solar collector as a function of absorber surface temperature.

b) Glass properties

1. Emissivity at 0-200°F	0.9
2. Specific heat at 212°F 32-212F	0.205
3. Thermal stress endurance (degrees differential)	450°F
4. Maximum working temperature (over entire glass area):	
Long exposure (greater than 100 hours)	450°F to 500°F
Short exposure (less than 10 hours)	500°F to 550°F

gure 8 - Thermal Performance Graph



3. Wind Load: The $34^{3}/_{16} \times 76^{3}/_{16} - {}^{1}/_{8}$ inch HERCULITE insulating unit for the PPG Baseline Collector can withstand a one-minute, fastest mile uniform wind load of 225 PSF (or 170 MPH) with a *probability of breakage equal to 8 lights per 1000.*

Snow Load: The $34^{3}/_{16} \times 76^{3}/_{16} - {}^{1}/_{8}$ inch HERCULITE insulating unit for the PPG Baseline Collector can withstand a long-term (more than one hour) uniform snow load of 170 PSF, with a *probability of breakage equal to 8 lights per 1000.*

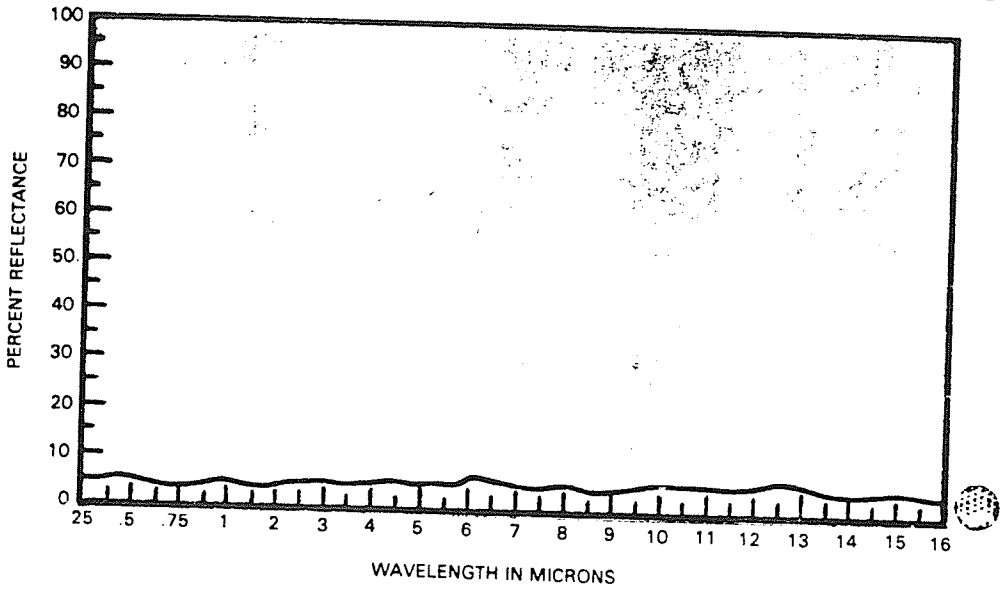
Hail: PPG $^{1}/_{8}$ inch HERCULITE glass has been shown to be effective against hailstone impact.

Absorbing surface — PPG's Baseline Solar Collector utilizes a flat black coating for the collector plate. The coating is PPG's DURACRON® Super 600 L/G (UC 9437) product. The DURACRON coating has a solar radiation absorbance, α, of 0.95 and an infrared emittance, ϵ, of 95 (α/ϵ 1.00) Figure 9, above, shows the percent reflectance of the DURACRON absorbing coating.

Collector Absorber Plates — PPG's Baseline Solar Collector utilizes a "Rolled and" 0.060 aluminum (1100 type) collector absorber plate.

Note: PPG Baseline Collectors may be constructed using copper or steel collector plates upon special order negotiated with PPG.)

Figure 9 - Spectral Reflectance Curve for PPG's DURACRON Super 600 L/G Black Paint



8. Insulating Backing — PPG's Baseline Collector utilizes fiber glass insulation backing.

9. Edge Retaining System — PPG's Baseline Solar Collector utilizes a special edge retaining system where:

a. The $^{1}/_{8}$ inch HERCULITE tempered glass plates are separated by a spacer. These spacers contain a desiccant which reduces condensation on the glass plates. This special design eliminates the effect of pressure build up with temperature changes.

b. The edge attachment system for the unit consists of a metal channel with high temperature resisting sealants that have the capability to withstand high temperature "no load" conditions.

FIELD TEST DATA

Four PPG Baseline Solar Collectors were installed in Melbourne, Florida. (See pictures, page 7.) The Solar Collector installation was designed to provide water heating. Figure 3, page 3, shows the schematic diagram for this installation.

Test data revealed a high correlation between experimental and theoretical collector efficiencies (see Figure 12).

gure 10-Base-line Solar Collector Efficiency $T_{AIR} = 50°F$

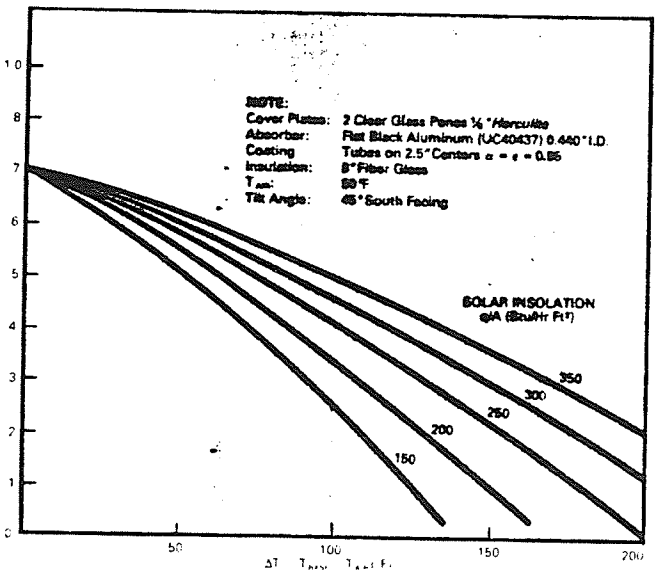


Figure 11-Base-line Solar Collector Efficiency $T_{AIR} = 100°F$

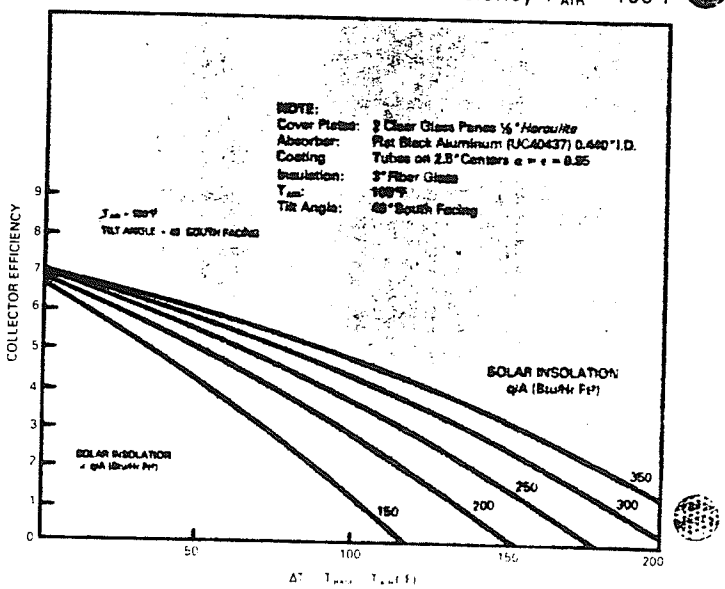
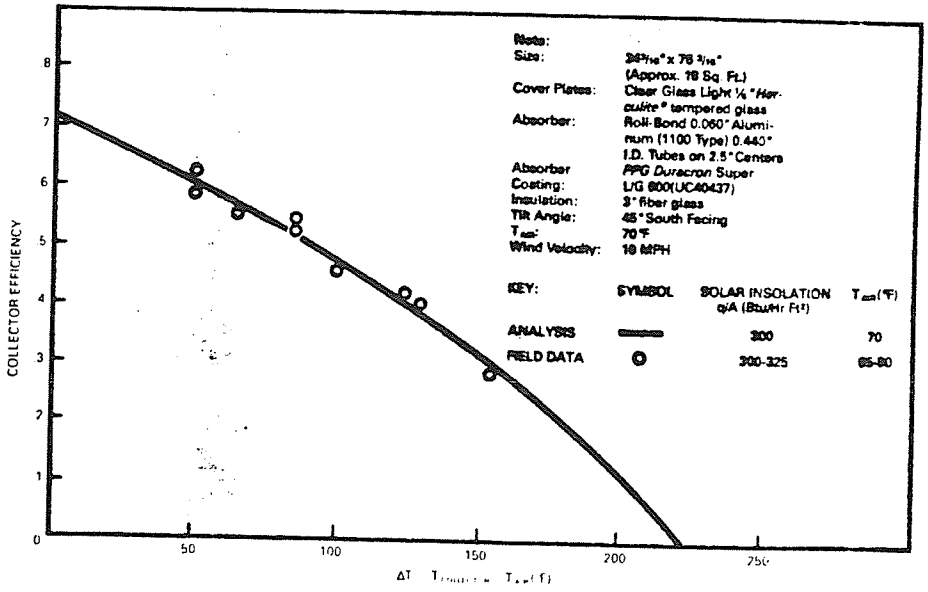


Figure 12-Comparison of Experimental and Theoretical Results for PPG Baseline Collector



APPENDIX V

INPUT AND OUTPUT DATA

This Appendix contains listings in the files that are required in order to run the finite difference program. The first file, RUNDATA.DAT, illustrate the data required to instruct the program to produce ten simulations on day 77337 for different controllers and flow rates. File CLCTRDAT.DAT is a listing of the node and conductor data that represents the physical system with a network of lumped nodes. Files B77231.DAT to B78007.DAT are the boundary conditions required to drive the network. The final listing is a sample output that can be obtained calling subroutines HPRINT, TPRINT, QPRINT, GPRINT and CPRINT in subroutines PRNOUT.

FILE: RUNDATA.DAT

```
77337F5,7.0,17.0,500.0,2
77337F4,7.0,17.0,250.0,2
77337F3,7.0,17.0,125.0,2
77337F2,7.0,17.0,62.50,2
77337F1,7.0,17.0,31.25,2
77337V5,7.0,17.0,500.0,2
77337V4,7.0,17.0,250.0,2
77337V3,7.0,17.0,125.0,2
77337V2,7.0,17.0,62.50,2
77337V1,7.0,17.0,31.25,2
END
```

```
C ****************** FILE: CLCTRDAT.DAT **************
C
C ************** NODE DATA BLOCK ********************
C
C NODE,TEMPERATURE,CAPACITANCE,HEAT SOURCE/SINK,AREA
C  I5 ,     F10.4 ,    F10.4  ,       F10.4    , F10.4
C
C COLLECTOR PLATE NODES
C
1,80.0,1.310E-02,0.0,5.43
2,80.0,2.0441E-02,0.0,8.125
3,80.0,2.0441E-02,0.0,8.125
4,80.0,1.533E-02,0.0,6.09
C
5,80.0,5.113E-02,0.0,20.3
6,80.0,7.645E-02,0.0,30.4
7,80.0,7.645E-02,0.0,30.4
8,80.0,5.734E-02,0.0,22.8
C
9,80.0,5.113E-02,0.0,20.3
10,80.0,7.645E-02,0.0,30.4
11,80.0,7.645E-02,0.0,30.4
12,80.0,5.734E-02,0.0,22.8
C
13,80.0,5.113E-02,0.0,20.3
14,80.0,7.645E-02,0.0,30.4
15,80.0,7.645E-02,0.0,30.4
16,80.0,5.734E-02,0.0,22.8
C
17,80.0,5.113E-02,0.0,20.3
18,80.0,7.645E-02,0.0,30.4
19,80.0,7.645E-02,0.0,30.4
20,80.0,5.734E-02,0.0,22.8
C
21,80.0,5.113E-02,0.0,20.3
22,80.0,7.645E-02,0.0,30.4
23,80.0,7.645E-02,0.0,30.4
24,80.0,5.734E-02,0.0,22.8
C
25,80.0,5.113E-02,0.0,20.3
26,80.0,7.645E-02,0.0,30.4
27,80.0,7.645E-02,0.0,30.4
28,80.0,5.734E-02,0.0,22.8
C
29,80.0,5.113E-02,0.0,20.3
30,80.0,7.645E-02,0.0,30.4
31,80.0,7.645E-02,0.0,30.4
32,80.0,5.734E-02,0.0,22.8
C
33,80.0,5.113E-02,0.0,20.3
```

```
34,80.0,7.645E-02,0.0,30.4
35,80.0,7.645E-02,0.0,30.4
36,80.0,5.734E-02,0.0,22.8
C
37,80.0,5.113E-02,0.0,20.3
38,80.0,7.645E-02,0.0,30.4
39,80.0,7.645E-02,0.0,30.4
40,80.0,5.734E-02,0.0,22.8
C
41,80.0,5.113E-02,0.0,20.3
42,80.0,7.645E-02,0.0,30.4
43,80.0,7.645E-02,0.0,30.4
44,80.0,5.734E-02,0.0,22.8
C
45,80.0,5.113E-02,0.0,20.3
46,80.0,7.645E-02,0.0,30.4
47,80.0,7.645E-02,0.0,30.4
48,80.0,5.734E-02,0.0,22.8
C
49,80.0,5.113E-02,0.0,20.3
50,80.0,7.645E-02,0.0,30.4
51,80.0,7.645E-02,0.0,30.4
52,80.0,5.734E-02,0.0,22.8
C
53,80.0,1.310E-02,0.0,5.43
54,80.0,2.0441E-02,0.0,8.125
55,80.0,2.0441E-02,0.0,8.125
56,80.0,1.533E-02,0.0,6.09
C
C TUBE NODES
C
57,80.0,0.00948,0.0,0.0
58,80.0,0.0442,0.0,0.0
59,80.0,0.0589,0.0,0.0
60,80.0,0.0543,0.0,0.0
C
61,80.0,0.03337,0.0,0.0
62,80.0,0.06674,0.0,0.0
63,80.0,0.06674,0.0,0.0
64,80.0,0.05006,0.0,0.0
C
65,80.0,0.03337,0.0,0.0
66,80.0,0.06674,0.0,0.0
67,80.0,0.06674,0.0,0.0
68,80.0,0.05006,0.0,0.0
C
69,80.0,0.03337,0.0,0.0
70,80.0,0.06674,0.0,0.0
71,80.0,0.06674,0.0,0.0
72,80.0,0.05006,0.0,0.0
```

```
C
73,80.0,0.03337,0.0,0.0
74,80.0,0.06674,0.0,0.0
75,80.0,0.06674,0.0,0.0
76,80.0,0.05006,0.0,0.0
C
77,80.0,0.03337,0.0,0.0
78,80.0,0.06674,0.0,0.0
79,80.0,0.06674,0.0,0.0
80,80.0,0.05006,0.0,0.0
C
81,80.0,0.03337,0.0,0.0
82,80.0,0.06674,0.0,0.0
83,80.0,0.06674,0.0,0.0
84,80.0,0.05006,0.0,0.0
C
85,80.0,0.03337,0.0,0.0
86,80.0,0.06674,0.0,0.0
87,80.0,0.06674,0.0,0.0
88,80.0,0.05006,0.0,0.0
C
89,80.0,0.03337,0.0,0.0
90,80.0,0.06674,0.0,0.0
91,80.0,0.06674,0.0,0.0
92,80.0,0.05006,0.0,0.0
C
93,80.0,0.03337,0.0,0.0
94,80.0,0.06674,0.0,0.0
95,80.0,0.06674,0.0,0.0
96,80.0,0.05006,0.0,0.0
C
97,80.0,0.03337,0.0,0.0
98,80.0,0.06674,0.0,0.0
99,80.0,0.06674,0.0,0.0
100,80.0,0.05006,0.0,0.0
C
101,80.0,0.00948,0.0,0.0
102,80.0,0.0442,0.0,0.0
103,80.0,0.0589,0.0,0.0
104,80.0,0.0543,0.0,0.0
C
C FIRST COVER PLATE
C
105,80.0,0.267,0.0,114.9
106,80.0,0.281,0.0,120.6
107,80.0,0.236,0.0,101.4
108,80.0,0.237,0.0,106.4
109,80.0,0.236,0.0,101.4
110,80.0,0.247,0.0,106.4
111,80.0,0.236,0.0,101.4
```

```
112,80.0,0.247,0.0,106.4
113,80.0,0.236,0.0,101.4
114,80.0,0.247,0.0,106.4
115,80.0,0.267,0.0,114.9
116,80.0,0.281,0.0,120.6
C
C SECOND COVER PLATE
C
117,80.0,0.790,0.0,443.4
118,80.0,0.967,0.0,415.6
119,80.0,0.790,0.0,443.4
C
C INLET BOUNDARY NODE
C  I.E. THE STORAGE TANK WITH 450 LBS OF WATER
C
120,40.0,450.0,0.0,0.0
C
C OUTLET DIFFUSION NODE
C
121,80.0,1.0,0.0,0.0
C
C ATTIC NODE
C
122,65.0,-1.0,0.0,0.0
C
C AMBIENT BOUNDARY NODE
C
123,80.0,-1.0,0.0,0.0
C
C EQUIVALENT SKY TEMPERATURE,BOUNDARY NODE.
C
124,65.0,-1.0,0.0,0.0
END
C
C *************** CONDUCTOR DATA BLOCK ************
C
C NA,NB,G( )
C
C I5,I5,F10.4
C
C
C LONGITUDINAL LINEAR CONDUCTORS
C
C TOP AND BOTTOM EDGES
C
1,5,1.111
53,49,1.111
C
2,6,1.662
54,50,1.662
```

```
C
3,7,1.662
55,51,1.662
C
4,8,1.247
56,52,1.247
C
C EDGE COLUMN
C
5,9,0.7044
9,13,0.7044
13,17,0.7044
17,21,0.7044
21,25,0.7044
25,29,0.7044
29,33,0.7044
33,37,0.7044
37,41,0.7044
41,45,0.7044
45,49,0.7044
C
C FIRST CENTER COLUMN
C
6,10,1.053
10,14,1.053
14,18,1.053
18,22,1.053
22,26,1.053
26,30,1.053
30,34,1.053
34,38,1.053
38,42,1.053
42,46,1.053
46,50,1.053
C
C SECOND CENTER COLUMN
C
7,11,1.053
11,15,1.053
15,19,1.053
19,23,1.053
23,27,1.053
27,31,1.053
31,35,1.053
35,39,1.053
39,43,1.053
43,47,1.053
47,51,1.053
C
C CENTER
```

```
C
8,12,0.7899
12,16,0.7899
16,20,0.7899
20,24,0.7899
24,28,0.7899
28,32,0.7899
32,36,0.7899
36,40,0.7899
40,44,0.7899
44,48,0.7899
48,52,0.7899
C
C LATERAL CONDUCTORS
C
C EDGES
C
1,2,0.499
2,3,0.499
3,4,0.499
C
53,54,0.499
54,55,0.499
55,56,0.499
C
C
5,6,1.865
13,14,1.865
21,22,1.865
29,30,1.865
37,38,1.865
45,46,1.865
C
6,7,1.556
14,15,1.556
22,23,1.556
30,31,1.556
38,39,1.556
46,47,1.556
C
7,8,1.778
15,16,1.778
23,24,1.778
31,32,1.778
39,40,1.778
47,48,1.778
C
9,10,1.865
17,18,1.865
25,26,1.865
```

```
33,34,1.865
41,42,1.865
49,50,1.865
C
10,11,1.556
18,19,1.556
26,27,1.556
34,35,1.556
42,43,1.556
50,51,1.556
C
11,12,1.778
19,20,1.778
27,28,1.778
35,36,1.778
43,44,1.778
51,52,1.778
C
C COLLECTOR PLATE NODES TO ATTIC (121)
C
1,122,3.033E-03
53,122,3.033E-03
C
2,122,4.514E-03
3,122,4.514E-03
54,122,4.514E-03
55,122,4.514E-03
C
4,122,3.383E-03
56,122,3.383E-03
C
5,122,1.127E-02
9,122,1.127E-02
13,122,1.127E-02
17,122,1.127E-02
21,122,1.127E-02
25,122,1.127E-02
29,122,1.127E-02
33,122,1.127E-02
37,122,1.127E-02
41,122,1.127E-02
45,122,1.127E-02
49,122,1.127E-02
C
6,122,1.688E-02
10,122,1.688E-02
14,122,1.688E-02
18,122,1.688E-02
22,122,1.688E-02
26,122,1.688E-02
```

```
30,122,1.688E-02
34,122,1.688E-02
38,122,1.688E-02
42,122,1.688E-02
46,122,1.688E-02
50,122,1.688E-02
7,122,1.688E-02
11,122,1.688E-02
15,122,1.688E-02
19,122,1.688E-02
23,122,1.688E-02
27,122,1.688E-02
31,122,1.688E-02
35,122,1.688E-02
39,122,1.688E-02
43,122,1.688E-02
47,122,1.688E-02
51,122,1.688E-02
C
8,122,1.266E-02
12,122,1.266E-02
16,122,1.266E-02
20,122,1.266E-02
24,122,1.266E-02
28,122,1.266E-02
32,122,1.266E-02
36,122,1.266E-02
40,122,1.266E-02
44,122,1.266E-02
48,122,1.266E-02
52,122,1.266E-02
C
C CONDUCTORS FROM COLLECTOR PLATE TO AMBIENT
C
5,123,0.01
9,123,0.01
13,123,0.01
17,123,0.01
21,123,0.01
25,123,0.01
29,123,0.01
33,123,0.01
37,123,0.01
41,123,0.01
45,123,0.01
49,123,0.01
C
1,123,0.01
53,123,0.01
2,123,0.01
```

```
3,123,0.01
4,123,0.01
54,123,0.01
55,123,0.01
56,123,0.01
C
C FLUID CONDUCTORS; ACTUAL VALUES CALCULATED IN PRE
C AS MASS FLOW RATE * CF
C
C -(DOWNSTREAM NODE),(UPSTREAM),0.0
C
C CONNECT INLET TO INLET MANIFOLD
C
-120,60,0.3
C
C CONNECT OUTLET TO OUTLET MANIFOLD
C
-104,121,0.3
C
C CONNECT OUTLET TO TANK (INLET NODE)
C
-121,120,0.3
C
C CONNECT REMAINING FLUID CONDUCTORS
C
-60,64,0.3
-60,59,0.3
-59,63,0.3
-59,58,0.3
-58,62,0.3
-58,57,0.3
-57,61,0.3
C
-61,65,0.3
-62,66,0.3
-63,67,0.3
-64,68,0.3
C
-65,69,0.3
-66,70,0.3
-67,71,0.3
-68,72,0.3
C
-69,73,0.3
-70,74,0.3
-71,75,0.3
-72,76,0.3
C
-73,77,0.3
-74,78,0.3
```

```
-75,79,0.3
-76,80,0.3
C
-77,81,0.3
-78,82,0.3
-79,83,0.3
-80,84,0.3
C
-81,85,0.3
-82,86,0.3
-83,87,0.3
-84,88,0.3
C
-85,89,0.3
-86,90,0.3
-87,91,0.3
-88,92,0.3
C
-89,93,0.3
-90,94,0.3
-91,95,0.3
-92,96,0.3
C
-93,97,0.3
-94,98,0.3
-95,99,0.3
-96,100,0.3
C
-97,101,0.3
-98,102,0.3
-99,103,0.3
-100,104,0.3
C
-101,102,0.3
-102,103,0.3
-103,104,0.3
C
C CONNECT PLATE TO TUBE NODES. ACTUAL CONDUCTOR VALUES ARE CALCULATED
C IN PRE VIA NUSSELT NUMBER
C
5,57,0.4
6,58,0.4
7,59,0.4
8,60,0.4
C
9,61,0.4
10,62,0.4
11,63,0.4
12,64,0.4
C
```

```
13,65,0.4
14,66,0.4
15,67,0.4
16,68,0.4
C
17,69,0.4
18,70,0.4
19,71,0.4
20,72,0.4
C
21,73,0.4
22,74,0.4
23,75,0.4
24,76,0.4
C
25,77,0.4
26,78,0.4
27,79,0.4
28,80,0.4
C
29,81,0.4
30,82,0.4
31,83,0.4
32,84,0.4
C
33,85,0.4
34,86,0.4
35,87,0.4
36,88,0.4
C
37,89,0.4
38,90,0.4
39,91,0.4
40,92,0.4
C
41,93,0.4
42,94,0.4
43,95,0.4
44,96,0.4
C
45,97,0.4
46,98,0.4
47,99,0.4
48,100,0.4
C
49,101,0.4
50,102,0.4
51,103,0.4
52,104,0.4
C
```

```
C COLLECTOR PLATE TO GLASS PLATE CONVECTION CONDUCTORS
C ACTUAL VALUES SET IN PRE
C
1,105,0.1
2,105,0.1
5,105,0.1
6,105,0.1
9,105,0.1
10,105,0.1
C
3,106,0.1
4,106,0.1
7,106,0.1
8,106,0.1
11,106,0.1
12,106,0.1
C
13,107,0.1
14,107,0.1
17,107,0.1
18,107,0.1
C
15,108,0.1
16,108,0.1
19,106,0.1
20,108,0.1
C
21,109,0.1
22,109,0.1
25,109,0.1
26,109,0.1
C
23,110,0.1
24,110,0.1
27,110,0.1
28,110,0.1
C
29,111,0.1
30,111,0.1
33,111,0.1
34,111,0.1
C
31,112,0.1
32,112,0.1
35,112,0.1
36,112,0.1
C
37,113,0.1
38,113,0.1
41,113,0.1
```

```
42,113,0.1
C
39,114,0.1
40,114,0.1
43,114,0.1
44,114,0.1
C
45,115,0.1
46,115,0.1
49,115,0.1
50,115,0.1
53,115,0.1
54,115,0.1
C
47,116,0.1
48,116,0.1
51,116,0.1
52,116,0.1
55,116,0.1
56,116,0.1
C
C FIRST COVER PLATE TO SECOND COVER PLATE CONVECTION CONDUCTORS
C ACTUAL VALUES SET IN PRE
C
105,117,0.1
106,117,0.1
107,117,0.1
108,117,0.1
C
109,118,0.1
110,118,0.1
111,118,0.1
112,118,0.1
C
113,119,0.1
114,119,0.1
115,119,0.1
116,119,0.1
C
C SECOND COVER PLATE TO AMBIENT CONVECTION CONDUCTORS
C ACTUAL VALUE SET IN PRE
C
117,123,0.1
118,123,0.1
119,123,0.1
C
C ********** RADIATION CONDUCTORS **********
C
C ACTUAL VALUES SET IN MAIN PROGRAM
C
```

```
C
C COLLECTOR PLATE TO GLASS PLATE RADIATION CONDUCTORS
C ACTUAL VALUES SET IN PRE
C
1,105,-0.1E-08
2,105,-0.1E-08
5,105,-0.1E-08
6,105,-0.1E-08
9,105,-0.1E-08
10,105,-0.1E-08
C
3,106,-0.1E-08
4,106,-0.1E-08
7,106,-0.1E-08
8,106,-0.1E-08
11,106,-0.1E-08
12,106,-0.1E-08
C
13,107,-0.1E-08
14,107,-0.1E-08
17,107,-0.1E-08
18,107,-0.1E-08
C
15,108,-0.1E-08
16,108,-0.1E-08
19,106,-0.1E-08
20,108,-0.1E-08
C
21,109,-0.1E-08
22,109,-0.1E-08
25,109,-0.1E-08
26,109,-0.1E-08
C
23,110,-0.1E-08
24,110,-0.1E-08
27,110,-0.1E-08
28,110,-0.1E-08
C
29,111,-0.1E-08
30,111,-0.1E-08
33,111,-0.1E-08
34,111,-0.1E-08
C
31,112,-0.1E-08
32,112,-0.1E-08
35,112,-0.1E-08
36,112,-0.1E-08
C
37,113,-0.1E-08
38,113,-0.1E-08
```

```
41,113,-0.1E-08
42,113,-0.1E-08
C
39,114,-0.1E-08
40,114,-0.1E-08
43,114,-0.1E-08
44,114,-0.1E-08
C
45,115,-0.1E-08
46,115,-0.1E-08
49,115,-0.1E-08
50,115,-0.1E-08
53,115,-0.1E-08
54,115,-0.1E-08
C
47,116,-0.1E-08
48,116,-0.1E-08
51,116,-0.1E-08
52,116,-0.1E-08
55,116,-0.1E-08
56,116,-0.1E-08
C
C FIRST COVER PLATE TO SECOND COVER PLATE RADIATION CONDUCTORS
C ACTUAL VALUES SET IN PRE
C
105,117,-0.1E-08
106,117,-0.1E-08
107,117,-0.1E-08
108,117,-0.1E-08
C
109,118,-0.1E-08
110,118,-0.1E-08
111,118,-0.1E-08
112,118,-0.1E-08
C
113,119,-0.1E-08
114,119,-0.1E-08
115,119,-0.1E-08
116,119,-0.1E-08
C
C SECOND COVER PLATE TO AMBIENT RADIATION CONDUCTORS
C ACTUAL VALUE SET IN PRE
C
117,123,-0.1E-08
118,123,-0.1E-08
119,123,-0.1E-08
END
```

```
C
C ****************** FILE: B77231.DAT ******************
C
C DAY 77231 -- AUG 19,1977
C LOCAL TIME (DECIMAL), SOLAR INSOLATION (WATTS/SQ M)
C
6.0,0.0
6.5,7.0
7.0,63.0
7.5,154.0
8.0,251.0
9.0,459.0
9.5,547.0
10.0,630.0
10.5,696.0
11.0,754.0
11.5,782.0
12.0,810.0
12.5,869.0
13.0,893.0
13.5,814.0
14.0,210.0
14.5,754.0
15.0,604.0
15.5,570.0
16.0,497.0
16.5,356.0
17.0,255.0
17.5,152.0
18.0,71.0
18.5,60.0
19.0,48.0
19.5,35.0
20.0,22.0
20.5,5.0
21.0,0.0
END
C
C LOCAL TIME (DECIMAL), AMBIENT (CELSIUS)
C SPEED (KM/HR), DIRECTION
C
6.0,11.0,9.0,S
7.0,11.0,9.0,S
8.0,12.8,9.0,S
9.0,15.0,13.0,S
10.0,17.6,15.0,S
11.0,18.8,13.0,SSE
12.0,21.7,17.0,S
13.0,22.3,11.0,SSW
14.0,23.0,11.0,SSE
```

```
15.0,23.6,2.0,NNW
16.0,23.1,22.0,NW
17.0,21.4,2.0,N
18.0,21.6,20.0,SE
19.0,17.6,15.0,SSE
20.0,16.4,19.0,SSW
21.0,15.9,15.0,S
END
C
C
C ****************** FILE: B77337.DAT ******************
C
C DAY 77337 -- DEC 3, 1977
C LOCAL TIME (DECIMAL), SOLAR INSOLATION (WATTS/SQ M)
C
7.0,0.0
7.25,0.0
7.5,0.0
7.75,1.0
8.0,4.0
8.25,14.0
8.5,214.0
8.75,325.0
9.0,397.0
9.25,412.0
9.5,334.0
9.75,301.0
10.0,579.0
10.25,424.0
10.5,551.0
10.75,703.0
11.0,602.0
11.25,796.0
11.50,689.0
11.75,551.0
12.0,708.0
12.25,695.0
12.50,799.0
12.75,785.0
13.00,582.0
13.25,742.0
13.5,701.0
13.75,662.0
14.0,619.0
14.25,568.0
14.50,512.0
14.75,445.0
15.0,395.0
15.25,331.0
15.5,259.0
```

```
15.75,192.0
16.0,62.0
16.25,17.0
16.5,3.0
16.75,1.0
17.0,0.0
END
C
C LOCAL TIME DECIMAL), AMBIENET (CELSIUS)
C SPEED (KM/HR), DIRECTION
C
7.0,-22.8,13.0,WNW
8.0,-22.1,17.0,NW
9.0,-23.1,15.0,NW
10.0,-23.3,11.0,NW
11.0,-22.3,15.0,NNW
12.0,-21.6,19.0,NW
13.0,-20.8,17.0,NW
14.0,-20.2,15.0,NNW
15.0,-20.2,11.0,NNW
16.0,-20.1,15.0,NNW
17.0,-21.2,11.0,NNW
END
C
C
C ****************** FILE: B77347.DAT ****************
C
C DAY 77347 -- DEC 13,1977
C LOCAL TIME (DECIMAL),SOLAR INSOLATION (WATTS/SQ M)
C
7.0,0.0
7.5,1.0
7.75,1.0
8.0,1.0
8.25,2.0
8.5,5.0
8.75,9.0
9.0,15.0
9.25,23.0
9.5,38.0
9.75,47.0
10.0,63.0
10.25,114.0
10.5,271.0
10.75,273.0
11.0,425.0
11.25,626.0
11.5,742.0
11.75,777.0
12.0,773.0
```

```
12.25,765.0
12.5,750.0
12.75,713.0
13.0,663.0
13.25,667.0
13.5,639.0
13.75,607.0
14.0,569.0
14.25,482.0
14.5,450.0
14.75,387.0
15.0,331.0
15.25,280.0
15.5,219.0
15.75,158.0
16.0,95.0
16.25,26.0
16.5,3.0
16.75,1.0
17.0,0.0
END
C
C LOCAL TIME (DECIMAL), AMBIENT (CELSIUS),
C SPEED(KM/HR), DIRECTION
C
7.0,-11.9,15.0,NNW
8.0,-12.4,11.0,NW
9.0,-12.7,15.0,NW
10.0,-12.8,15.0,NNW
11.0,-12.8,15.0,NW
12.0,-13.0,11.0,NNW
13.0,-12.8,13.0,NW
14.0,-13.2,11.0,NNW
15.0,-13.2,7.0,NNW
16.0,-12.5,6.0,W
17.0,-14.5,7.0,WNW
END
C
C
C ****************** FILE: B77360.DAT ******************
C
C DAY 77360 -- DEC 26,1977
C LOCAL TIME (DECIMAL), SOLAR INSOLATION (WATTS/SQ M)
C
7.0,0.0
7.25,0.0
7.5,0.0
7.75,0.0
8.0,1.0
8.25,4.0
```

```
8.5,13.0
8.75,156.0
9.0,373.0
9.25,485.0
9.5,557.0
9.75,611.0
10.0,668.0
10.25,737.0
10.5,774.0
10.75,806.0
11.0,823.0
11.25,836.0
11.5,853.0
11.75,852.0
12.0,852.0
12.25,844.0
12.5,827.0
12.75,684.0
13.0,763.0
13.25,733.0
13.5,581.0
13.75,690.0
14.0,645.0
14.25,579.0
14.5,546.0
14.75,487.0
15.0,436.0
15.25,358.0
15.5,248.0
15.75,238.0
16.0,181.0
16.25,87.0
16.5,16.0
16.75,3.0
17.0,1.0
END
C
C LOCAL TIME (DECIMAL), AMBIENT (CELSIUS)
C SPEED (KM/HR), DIRECTION
C
7.0,-28.7,20.0,NW
8.0,-29.2,24.0,NW
9.0,-29.1,17.0,NW
10.0,-28.7,20.0,NW
11.0,-28.2,17.0,NW
12.0,-27.2,17.0,NW
13.0,-26.5,20.0,NW
14.0,-26.3,20.0,NNW
15.0,-25.8,19.0,NW
16.0,-26.0,20.0,NW
```

```
17.0,-27.5,17.0,NW
END
C
C
C ****************** FILE: B78001.DAT ******************
C
C DAY 78001 -- JAN 1,1978
C LOCAL TIME (DECIMAL), SOLAR INSOLATION (WATTS/SQ M)
C
8.0,0.0
8.25,9.0
8.5,22.0
8.75,251.0
9.0,195.0
9.25,149.0
9.5,176.0
9.75,205.0
10.0,468.0
10.25,356.0
10.5,358.0
10.75,418.0
11.0,282.0
11.25,226.0
11.5,223.0
11.75,149.0
12.0,129.0
12.25,104.0
12.5,68.0
12.75,70.0
13.0,66.0
13.25,63.0
13.5,50.0
13.75,51.0
14.0,56.0
14.25,52.0
14.5,40.0
14.75,36.0
15.0,26.0
15.25,20.0
15.5,16.0
15.75,13.0
16.0,9.0
16.25,4.0
16.5,2.0
16.75,0.0
17.0,0.0
END
C
C LOCAL TIME (DECIMAL), AMBIENT (CELSIUS)
C SPEED (KM/HR), DIRECTION
```

```
C
8.0,-17.4,19.0,SW
9.0,-18.7,24.0,SW
10.0,-18.0,28.0,SW
11.0,-17.4,26.0,SW
12.0,-15.8,30.0,SW
13.0,-14.2,24.0,SW
14.0,-11.8,26.0,WSW
15.0,-11.7,26.0,W
16.0,-10.8,24.0,WNW
17.0,-10.6,22.0,WNW
END
C
C
C ****************** FILE: B78002.DAT ******************
C
C DAY 78002 -- JAN 2,1978
C LOCAL TIME (DECIMAL), SOLAR INSOLATION (WATTS/SQ M)
C
8.0,0.0
8.25,0.0
8.5,3.0
8.75,10.0
9.0,25.0
9.25,406.0
9.5,544.0
9.75,613.0
10.0,689.0
10.25,670.0
10.5,659.0
10.75,744.0
11.0,371.0
11.25,852.0
11.5,852.0
11.75,373.0
12.0,687.0
12.25,711.0
12.5,827.0
12.75,752.0
13.0,808.0
13.5,749.0
13.75,714.0
14.0,672.0
14.25,575.0
14.5,525.0
14.75,560.0
15.0,492.0
15.25,398.0
15.5,323.0
15.75,261.0
```

```
16.0,186.0
16.25,19.0
16.5,22.0
16.75,3.0
17.0,0.0
END
C
C LOCAL TIME (DECIMAL), AMBIENT (CELSIUS)
C SPEED (KM/HR), DIRECTION
C
8.0,-10.8,20.0,WNW
9.0,-12.0,15.0,WNW
10.0,-12.3,22.0,NW
11.0,-11.9,22.0,NW
12.0,-11.5,19.0,NW
13.0,-11.0,17.0,W
14.0,-11.5,15.0,W
15.0,-11.0,15.0,WNW
16.0,-12.1,15.0,W
17.0,-12.1,15.0,W
END
C
C
C ****************** FILE: B78007.DAT ******************
C
C DAY 78007 -- JAN 7,1978
C LOCAL TIME (DECIMAL), SOLAR INSOLATION (WATTS/SQ M)
C
8.0,0.0
8.25,1.0
8.5,3.0
8.75,9.0
9.0,17.0
9.25,28.0
9.5,39.0
9.75,59.0
10.0,79.0
10.25,105.0
10.5,85.0
10.75,142.0
11.0,148.0
11.25,209.0
11.5,217.0
11.75,249.0
12.0,217.0
12.25,218.0
12.5,371.0
12.75,569.0
13.0,859.0
13.25,516.0
```

```
13.5,600.0
13.75,478.0
14.0,222.0
14.25,408.0
14.5,311.0
14.75,134.0
15.0,79.0
15.25,69.0
15.5,80.0
15.75,46.0
16.0,32.0
16.25,18.0
16.5,10.0
16.75,3.0
17.0,1.0
END
C
C LOCAL TIME (DECIMAL), AMBIENT (CELSIUS)
C SPEED (KM/HR), DIRECTION
C
8.0,-18.9,28.0,NW
9.0,-19.2,31.0,NW
10.0,-19.4,33.0,NW
11.0,-19.4,31.0,NNW
12.0,-19.4,31.0,NNW
13.0,-19.8,31.0,NW
14.0,-20.5,35.0,NW
15.0,-20.6,39.0,NW
16.0,-21.3,35.0,NNW
17.0,-22.1,30.0,NNW
END
```

PPG ALUMINUM COLLECTOR: TEST CASE

TIMEN = 12.10	LOOPCT= 1	DRLXCC= .1868E-01	DTIMEU= .2773E-02	OUTPUT= .1000E+00
TIMED = 12.10	NLOOP = 50	NDRLXC= 120	DTIMEI= .2778E-02	TIMEND= 17.00
TIMEM = 12.10	NAMSIZ= 9	DRLXCA= .5000E-01	NNOD = 124	NCOND = 413

NODE TEMPERATURES(NEW)

T(1)= 102.	T(2)= 98.9	T(3)= 96.7	T(4)= 95.3	T(5)= 98.2	T(6)= 93.5
T(7)= 90.9	T(8)= 89.3	T(9)= 96.6	T(10)= 95.9	T(11)= 94.8	T(12)= 93.8
T(13)= 97.5	T(14)= 96.9	T(15)= 95.8	T(16)= 94.8	T(17)= 98.5	T(18)= 97.9
T(19)= 97.0	T(20)= 95.8	T(21)= 99.6	T(22)= 98.9	T(23)= 97.9	T(24)= 97.0
T(25)= 101.	T(26)= 99.8	T(27)= 98.8	T(28)= 98.0	T(29)= 102.	T(30)= 101.
T(31)= 99.8	T(32)= 99.0	T(33)= 103.	T(34)= 102.	T(35)= 101.	T(36)= 99.9
T(37)= 104.	T(38)= 103.	T(39)= 102.	T(40)= 101.	T(41)= 105.	T(42)= 104.
T(43)= 103.	T(44)= 102.	T(45)= 106.	T(46)= 104.	T(47)= 103.	T(48)= 102.
T(49)= 109.	T(50)= 104.	T(51)= 102.	T(52)= 101.	T(53)= 112.	T(54)= 109.
T(55)= 107.	T(56)= 106.	T(57)= 89.0	T(58)= 88.1	T(59)= 87.2	T(60)= 86.6
T(61)= 90.3	T(62)= 89.0	T(63)= 88.1	T(64)= 87.6	T(65)= 91.4	T(66)= 90.0
T(67)= 89.0	T(68)= 88.6	T(69)= 92.6	T(70)= 90.9	T(71)= 89.9	T(72)= 89.5
T(73)= 93.7	T(74)= 91.9	T(75)= 90.9	T(76)= 90.6	T(77)= 94.9	T(78)= 92.8
T(79)= 91.8	T(80)= 91.6	T(81)= 96.0	T(82)= 93.7	T(83)= 92.8	T(84)= 92.6
T(85)= 97.1	T(86)= 94.7	T(87)= 93.7	T(88)= 93.8	T(89)= 98.2	T(90)= 95.6
T(91)= 94.7	T(92)= 94.6	T(93)= 99.2	T(94)= 96.6	T(95)= 95.6	T(96)= 95.5
T(97)= 100.	T(98)= 97.5	T(99)= 96.5	T(100)= 96.5	T(101)= 101.	T(102)= 99.6
T(103)= 98.8	T(104)= 98.6	T(105)= 61.2	T(106)= 63.4	T(107)= 61.9	T(108)= 55.2
T(109)= 63.1	T(110)= 62.3	T(111)= 64.2	T(112)= 63.3	T(113)= 66.1	T(114)= 65.1
T(115)= 67.6	T(116)= 65.7	T(117)= 25.7	T(118)= 26.1	T(119)= 27.8	T(120)= 86.4
T(121)= 98.6	T(122)= 28.6	T(123)= 8.62	T(124)= -2.38	T(

NODE HEAT SOURCE/SINK

Q(1)= 9.151	Q(2)= 13.69	Q(3)= 13.69	Q(4)= 10.26	Q(5)= 34.21	Q(6)= 51.23
Q(7)= 51.23	Q(8)= 38.43	Q(9)= 34.21	Q(10)= 51.23	Q(11)= 51.23	Q(12)= 38.43
Q(13)= 34.21	Q(14)= 51.23	Q(15)= 51.23	Q(16)= 38.43	Q(17)= 34.21	Q(18)= 51.23
Q(19)= 51.23	Q(20)= 38.43	Q(21)= 34.21	Q(22)= 51.23	Q(23)= 51.23	Q(24)= 38.43
Q(25)= 34.21	Q(26)= 51.23	Q(27)= 51.23	Q(28)= 38.43	Q(29)= 34.21	Q(30)= 51.23
Q(31)= 51.23	Q(32)= 38.43	Q(33)= 34.21	Q(34)= 51.23	Q(35)= 51.23	Q(36)= 38.43
Q(37)= 34.21	Q(38)= 51.23	Q(39)= 51.23	Q(40)= 38.43	Q(41)= 34.21	Q(42)= 51.23
Q(43)= 51.23	Q(44)= 38.43	Q(45)= 34.21	Q(46)= 51.23	Q(47)= 51.23	Q(48)= 38.43
Q(49)= 34.21	Q(50)= 51.23	Q(51)= 51.23	Q(52)= 38.43	Q(53)= 9.151	Q(54)= 13.69
Q(55)= 13.69	Q(56)= 10.26	Q(57)= 0.000	Q(58)= 0.000	Q(59)= 0.000	Q(60)= 0.000
Q(61)= 0.000	Q(62)= 0.000	Q(63)= 0.000	Q(64)= 0.000	Q(65)= 0.000	Q(66)= 0.000
Q(67)= 0.000	Q(68)= 0.000	Q(69)= 0.000	Q(70)= 0.000	Q(71)= 0.000	Q(72)= 0.000
Q(73)= 0.000	Q(74)= 0.000	Q(75)= 0.000	Q(76)= 0.000	Q(77)= 0.000	Q(78)= 0.000
Q(79)= 0.000	Q(80)= 0.000	Q(81)= 0.000	Q(82)= 0.000	Q(83)= 0.000	Q(84)= 0.000
Q(85)= 0.000	Q(86)= 0.000	Q(87)= 0.000	Q(88)= 0.000	Q(89)= 0.000	Q(90)= 0.000
Q(91)= 0.000	Q(92)= 0.000	Q(93)= 0.000	Q(94)= 0.000	Q(95)= 0.000	Q(96)= 0.000
Q(97)= 0.000	Q(98)= 0.000	Q(99)= 0.000	Q(100)= 0.000	Q(101)= 0.000	Q(102)= 0.000

Q(103)= 0.000	Q(104)= 0.000	Q(105)= 0.000	Q(106)= 0.000	Q(107)= 0.000	Q(108)= 0.000
Q(109)= 0.000	Q(110)= 0.000	Q(111)= 0.000	Q(112)= 0.000	Q(113)= 0.000	Q(114)= 0.000
Q(115)= 0.000	Q(116)= 0.000	Q(117)= 0.000	Q(118)= 0.000	Q(119)= 0.000	Q(120)= 0.000
Q(121)= 0.000	Q(122)= 0.000	Q(123)= 0.000	Q(124)= 0.000	Q(	

NODE CAPACITANCES

C(1)=	.1310E-01	C(2)=	.2044E-01	C(3)=	.2044E-01	C(4)=	.1533E-01	C(5)=	.5113E-01	C(6)=	.7645E-01
C(7)=	.7645E-01	C(8)=	.5734E-01	C(9)=	.5113E-01	C(10)=	.7645E-01	C(11)=	.7645E-01	C(12)=	.5734E-01
C(13)=	.5113E-01	C(14)=	.7645E-01	C(15)=	.7645E-01	C(16)=	.5734E-01	C(17)=	.5113E-01	C(18)=	.7645E-01
C(19)=	.7645E-01	C(20)=	.5734E-01	C(21)=	.5113E-01	C(22)=	.7645E-01	C(23)=	.7645E-01	C(24)=	.5734E-01
C(25)=	.5113E-01	C(26)=	.7645E-01	C(27)=	.7645E-01	C(28)=	.5734E-01	C(29)=	.5113E-01	C(30)=	.7645E-01
C(31)=	.7645E-01	C(32)=	.5734E-01	C(33)=	.5113E-01	C(34)=	.7645E-01	C(35)=	.7645E-01	C(36)=	.5734E-01
C(37)=	.5113E-01	C(38)=	.7645E-01	C(39)=	.7645E-01	C(40)=	.5734E-01	C(41)=	.5113E-01	C(42)=	.7645E-01
C(43)=	.7645E-01	C(44)=	.5734E-01	C(45)=	.5113E-01	C(46)=	.7645E-01	C(47)=	.7645E-01	C(48)=	.5734E-01
C(49)=	.5113E-01	C(50)=	.7645E-01	C(51)=	.7645E-01	C(52)=	.5734E-01	C(53)=	.1310E-01	C(54)=	.2044E-01
C(55)=	.2044E-01	C(56)=	.1533E-01	C(57)=	.9480E-02	C(58)=	.4420E-01	C(59)=	.5890E-01	C(60)=	.5430E-01
C(61)=	.3337E-01	C(62)=	.6674E-01	C(63)=	.6674E-01	C(64)=	.5006E-01	C(65)=	.3337E-01	C(66)=	.6674E-01
C(67)=	.6674E-01	C(68)=	.5006E-01	C(69)=	.3337E-01	C(70)=	.6674E-01	C(71)=	.6674E-01	C(72)=	.5006E-01
C(73)=	.3337E-01	C(74)=	.6674E-01	C(75)=	.6674E-01	C(76)=	.5006E-01	C(77)=	.3337E-01	C(78)=	.6674E-01
C(79)=	.6674E-01	C(80)=	.5006E-01	C(81)=	.3337E-01	C(82)=	.6674E-01	C(83)=	.6674E-01	C(84)=	.5006E-01
C(85)=	.3337E-01	C(86)=	.6674E-01	C(87)=	.6674E-01	C(88)=	.5006E-01	C(89)=	.3337E-01	C(90)=	.6674E-01
C(91)=	.6674E-01	C(92)=	.5006E-01	C(93)=	.3337E-01	C(94)=	.6674E-01	C(95)=	.6674E-01	C(96)=	.5006E-01
C(97)=	.3337E-01	C(98)=	.6674E-01	C(99)=	.6674E-01	C(100)=	.5006E-01	C(101)=	.9480E-02	C(102)=	.4420E-01
C(103)=	.5890E-01	C(104)=	.5430E-01	C(105)=	.2670	C(106)=	.2810	C(107)=	.2360	C(108)=	.2370
C(109)=	.2360	C(110)=	.2470	C(111)=	.2360	C(112)=	.2470	C(113)=	.2360	C(114)=	.2470
C(115)=	.2670	C(116)=	.2810	C(117)=	.7900	C(118)=	.9670	C(119)=	.7900	C(120)=	450.0
C(121)=	1.0000	C(122)=	-1.0000	C(123)=	-1.0000	C(124)=	-1.0000	C(			

NODE TO NODE CONDUCTORS

G#	NA	NB	VALUE	G#	NA	NB	VALUE	G#	NA	NB	VALUE	G#	NA	NB	VALUE
1	1	5	1.111	2	53	49	1.111	3	2	6	1.662	4	54	50	1.662
5	3	7	1.662	6	55	51	1.662	7	4	8	1.247	8	56	52	1.247
9	5	9	.7044	10	9	13	.7044	11	13	17	.7044	12	17	21	.7044
13	21	25	.7044	14	25	29	.7044	15	29	33	.7044	16	33	37	.7044
17	37	41	.7044	18	41	45	.7044	19	45	49	.7044	20	6	10	1.053
21	10	14	1.053	22	14	18	1.053	23	18	22	1.053	24	22	26	1.053
25	26	30	1.053	26	30	34	1.053	27	34	38	1.053	28	38	42	1.053
29	42	46	1.053	30	46	50	1.053	31	7	11	1.053	32	11	15	1.053
33	15	19	1.053	34	19	23	1.053	35	23	27	1.053	36	27	31	1.053
37	31	35	1.053	38	35	39	1.053	39	39	43	1.053	40	43	47	1.053
41	47	51	1.053	42	8	12	.7899	43	12	16	.7899	44	16	20	.7899
45	20	24	.7899	46	24	28	.7899	47	28	32	.7899	48	32	36	.7899
49	36	40	.7899	50	40	44	.7899	51	44	48	.7899	52	48	52	.7899
53	1	2	.4990	54	2	3	.4990	55	3	4	.4990	56	53	54	.4990
57	54	55	.4990	58	55	56	.4990	59	5	6	1.865	60	13	14	1.865
61	21	22	1.865	62	29	30	1.865	63	37	38	1.865	64	45	46	1.865

65	6	7	1.556	66	14	15	1.556	67	22	23	1.556	68	30	31	1.556
69	38	39	1.556	70	46	47	1.556	71	7	8	1.778	72	15	16	1.778
73	23	24	1.778	74	31	32	1.778	75	39	40	1.778	76	47	48	1.778
77	9	10	1.865	78	17	18	1.865	79	25	26	1.865	80	33	34	1.865
81	41	42	1.865	82	49	50	1.865	83	10	11	1.556	84	18	19	1.556
85	26	27	1.556	86	34	35	1.556	87	42	43	1.556	88	50	51	1.556
89	11	12	1.778	90	19	20	1.778	91	27	28	1.778	92	35	36	1.778
93	43	44	1.778	94	51	52	1.778	95	1	122	.3033E-02	96	53	122	.3033E-02
97	2	122	.4514E-02	98	3	122	.4514E-02	99	54	122	.4514E-02	100	55	122	.4514E-02
101	4	122	.3383E-02	102	56	122	.3383E-02	103	5	122	.1127E-01	104	9	122	.1127E-01
105	13	122	.1127E-01	106	17	122	.1127E-01	107	21	122	.1127E-01	108	25	122	.1127E-01
109	29	122	.1127E-01	110	33	122	.1127E-01	111	37	122	.1127E-01	112	41	122	.1127E-01
113	45	122	.1127E-01	114	49	122	.1127E-01	115	6	122	.1688E-01	116	10	122	.1688E-01
117	14	122	.1688E-01	118	18	122	.1688E-01	119	22	122	.1688E-01	120	26	122	.1688E-01
121	30	122	.1688E-01	122	34	122	.1688E-01	123	38	122	.1688E-01	124	42	122	.1688E-01
125	46	122	.1688E-01	126	50	122	.1688E-01	127	7	122	.1688E-01	128	11	122	.1688E-01
129	15	122	.1688E-01	130	19	122	.1688E-01	131	23	122	.1688E-01	132	27	122	.1688E-01
133	31	122	.1688E-01	134	35	122	.1688E-01	135	39	122	.1688E-01	136	43	122	.1688E-01
137	47	122	.1688E-01	138	51	122	.1688E-01	139	8	122	.1266E-01	140	12	122	.1266E-01
141	16	122	.1266E-01	142	20	122	.1266E-01	143	24	122	.1266E-01	144	28	122	.1266E-01
145	32	122	.1266E-01	146	36	122	.1266E-01	147	40	122	.1266E-01	148	44	122	.1266E-01
149	48	122	.1266E-01	150	52	122	.1266E-01	151	5	123	.1000E-01	152	9	123	.1000E-01
153	13	123	.1000E-01	154	17	123	.1000E-01	155	21	123	.1000E-01	156	25	123	.1000E-01
157	29	123	.1000E-01	158	33	123	.1000E-01	159	37	123	.1000E-01	160	41	123	.1000E-01
161	45	123	.1000E-01	162	49	123	.1000E-01	163	1	123	.1000E-01	164	53	123	.1000E-01
165	2	123	.1000E-01	166	3	123	.1000E-01	167	4	123	.1000E-01	168	54	123	.1000E-01
169	55	123	.1000E-01	170	56	123	.1000E-01	171	-120	60	250.0	172	-104	121	250.0
173	-121	120	250.0	174	-60	64	28.85	175	-60	59	96.15	176	-59	63	38.46
177	-59	58	57.69	178	-58	62	38.46	179	-58	57	19.23	180	-57	61	19.23
181	-61	65	19.23	182	-62	66	38.46	183	-63	67	38.46	184	-64	68	28.85
185	-65	69	19.23	186	-66	70	38.46	187	-67	71	38.46	188	-68	72	28.85
189	-69	73	19.23	190	-70	74	38.46	191	-71	75	38.46	192	-72	76	28.85
193	-73	77	19.23	194	-74	78	38.46	195	-75	79	38.46	196	-76	80	28.85
197	-77	81	19.23	198	-78	82	38.46	199	-79	83	38.46	200	-80	84	28.85
201	-81	85	19.23	202	-82	86	38.46	203	-83	87	38.46	204	-84	88	28.85
205	-85	89	19.23	206	-86	90	38.46	207	-87	91	38.46	208	-88	92	28.85
209	-89	93	19.23	210	-90	94	38.46	211	-91	95	38.46	212	-92	96	28.85
213	-93	97	19.23	214	-94	98	38.46	215	-95	99	38.46	216	-96	100	28.85
217	-97	101	19.23	218	-98	102	38.46	219	-99	103	38.46	220	-100	104	28.85
221	-101	102	19.23	222	-102	103	57.69	223	-103	104	96.15	224	5	57	1.867
225	6	58	10.16	226	7	59	14.49	227	8	60	16.55	228	9	61	3.792
229	10	62	5.203	230	11	63	5.203	231	12	64	4.556	232	13	65	3.792
233	14	66	5.203	234	15	67	5.203	235	16	68	4.556	236	17	69	3.792
237	18	70	5.203	238	19	71	5.203	239	20	72	4.556	240	21	73	3.792
241	22	74	5.203	242	23	75	5.203	243	24	76	4.556	244	25	77	3.792
245	26	78	5.203	246	27	79	5.203	247	28	80	4.556	248	29	81	3.792
249	30	82	5.203	250	31	83	5.203	251	32	84	4.556	252	33	85	3.792
253	34	86	5.203	254	35	87	5.203	255	36	88	4.556	256	37	89	3.792

257	38	90	5.203	258	39	91	5.203	259	40	92	4.556	260	41	93	3.792
261	42	94	5.203	262	43	95	5.203	263	44	96	4.556	264	45	97	3.792
265	46	98	5.203	266	47	99	5.203	267	48	100	4.556	268	49	101	1.867
269	50	102	10.16	270	51	103	14.49	271	52	104	16.55	272	1	105	.2524E-01
273	2	105	.3736E-01	274	5	105	.9310E-01	275	6	105	.1367	276	9	105	.9250E-01
277	10	105	.1381	278	3	106	.3670E-01	279	4	106	.2734E-01	280	7	106	.1337
281	8	106	.9939E-01	282	11	106	.1362	283	12	106	.1016	284	13	107	.9256E-01
285	14	107	.1383	286	17	107	.9295E-01	287	18	107	.1389	288	15	108	.1414
289	16	108	.1056	290	19	106	.1375	291	20	108	.1060	292	21	109	.9288E-01
293	22	109	.1387	294	25	109	.9326E-01	295	26	109	.1392	296	23	110	.1387
297	24	110	.1036	298	27	110	.1392	299	28	110	.1040	300	29	111	.9326E-01
301	30	111	.1392	302	33	111	.9362E-01	303	34	111	.1397	304	31	112	.1391
305	32	112	.1040	306	35	112	.1396	307	36	112	.1044	308	37	113	.9333E-01
309	38	113	.1392	310	41	113	.9369E-01	311	42	113	.1397	312	39	114	.1392
313	40	114	.1041	314	43	114	.1396	315	44	114	.1044	316	45	115	.9359E-01
317	46	115	.1393	318	49	115	.9466E-01	319	50	115	.1393	320	53	115	.2560E-01
321	54	115	.3795E-01	322	47	116	.1397	323	48	116	.1044	324	51	116	.1391
325	52	116	.1038	326	55	116	.3792E-01	327	56	116	.2830E-01	328	105	117	.6220
329	106	117	.6577	330	107	117	.5502	331	108	117	.5632	332	109	118	.5518
333	110	118	.5773	334	111	118	.5538	335	112	118	.5793	336	113	119	.5540
337	114	119	.5795	338	115	119	.6308	339	116	119	.6581	340	117	123	7.692
341	118	123	7.692	342	119	123	7.692	343	1	105	-.6118E-10	344	2	105	-.9144E-10
345	5	105	-.2273E-09	346	6	105	-.3389E-09	347	9	105	-.2273E-09	348	10	105	-.3389E-09
349	3	106	-.9146E-10	350	4	106	-.6861E-10	351	7	106	-.3391E-09	352	8	106	-.2551E-09
353	11	106	-.3391E-09	354	12	106	-.2551E-09	355	13	107	-.2270E-09	356	14	107	-.3383E-09
357	17	107	-.2270E-09	358	18	107	-.3383E-09	359	15	108	-.3385E-09	360	16	108	-.2548E-09
361	19	106	-.3391E-09	362	20	108	-.2548E-09	363	21	109	-.2270E-09	364	22	109	-.3383E-09
365	25	109	-.2270E-09	366	26	109	-.3383E-09	367	23	110	-.3385E-09	368	24	110	-.2548E-09
369	27	110	-.3385E-09	370	28	110	-.2548E-09	371	29	111	-.2270E-09	372	30	111	-.3383E-09
373	33	111	-.2270E-09	374	34	111	-.3383E-09	375	31	112	-.3385E-09	376	32	112	-.2548E-09
377	35	112	-.3385E-09	378	36	112	-.2548E-09	379	37	113	-.2270E-09	380	38	113	-.3383E-09
381	41	113	-.2270E-09	382	42	113	-.3383E-09	383	39	114	-.3385E-09	384	40	114	-.2548E-09
385	43	114	-.3385E-09	386	44	114	-.2548E-09	387	45	115	-.2273E-09	388	46	115	-.3389E-09
389	49	115	-.2273E-09	390	50	115	-.3389E-09	391	53	115	-.6118E-10	392	54	115	-.9144E-10
393	47	116	-.3391E-09	394	48	116	-.2551E-09	395	51	116	-.3391E-09	396	52	116	-.2551E-09
397	55	116	-.9146E-10	398	56	116	-.6861E-10	399	105	117	-.1281E-08	400	106	117	-.1344E-08
401	107	117	-.1132E-08	402	108	117	-.1187E-08	403	109	118	-.1131E-08	404	110	118	-.1187E-08
405	111	118	-.1131E-08	406	112	118	-.1187E-08	407	113	119	-.1132E-08	408	114	119	-.1187E-08
409	115	119	-.1281E-08	410	116	119	-.1344E-08	411	117	123	-.5008E-08	412	118	123	-.4694E-08
413	119	123	-.5008E-08												