PFC/RR-87-11 DOE/ID-01570-2

LITFIRE USER'S GUIDE

D.S. Barnett, E. Yachimiak,
V. Gilberti, and M.S. Tillack

August 1987

Plasma Fusion Center
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 USA

*This work was supported by EG&G Idaho, Inc. and the
U.S. Department of Energy, Idaho Operations Office
under DOE Contract No. DE-AC07-76ID01570

LITFIRE USER'S GUIDE

Abstract

This document contains instructions for using the LITFIRE lithium fire simulation code in its form as of August 1987. The code is capable of simulating fires in a single compartment connected to an optional second compartment, or in an insulated pan suspended in one compartment. Lithium or lithium-lead eutectic may be used as the fuel, reacting with an atmosphere of oxygen, nitrogen, water vapor, or any mixture thereof. Any inert gas may be included in the compartment atmosphere and lithium only may be burned in a carbon dioxide atmosphere without oxygen. An option for liquid metal-concrete interaction exists as well as the following options for mitigating the effects of the fire: gas flooding, emergency space cooling, emergency floor cooling, aerosol removal and gas injection. The guide also includes the following:

- a description of the workings of the code, including the various options available to the user
- a description of the physics of lithium fires and heat and mass transfer
- instructions for running the code
- a list of sample input files
- a listing of the code with a glossary defining code variables

TABLE OF CONTENTS

LIST OF FIGURES

I. Introduction

LITFIRE is a computer code which simulates lithium fires in fusion reactors by generating the time histories of the temperature and pressure profiles occurring in a reactor containment in response to a lithium or lithium-lead eutectic spill and fire. The fire may take place in a single cell connected to an optional second cell, or in an insulated pan in a single cell. The lithium or lithium-lead may react with any mixture of oxygen, nitrogen or water vapor; an inert gas may also be included in the cell atmospheres. Lithium only may be burned in a carbon dioxide atmosphere without oxygen. An option for liquid metal-concrete interaction is available as well as the following options for mitigating the effects of a fire: gas flooding, emergency space cooling, emergency floor cooling, aerosol removal and gas injection. This user's guide includes a brief description of the physics of lithium fires, the workings of the code and the options available to users, and also includes a listing of the code with a complete glossary of variables used in LITFIRE.

The present version of LITFIRE is available on the PFCVAX at MIT and on the Crays at the National Magnetic Fusion Energy Computer Center (NMFECC, hereafter referred to as MFE) It is written in FORTRAN 77 and is compatible with the computing facilities at which it is located. The instructions present in this guide will enable a user to execute the code at either location.

Other information on LITFIRE such as more detailed descriptions of physical models or correlations used in the code are available in the references listed in the reference section of the guide.

II. Program Description

To assist the new user of LITFIRE in understanding the code, a brief descriptive section is included. The description of the code sections is presented in the order that they are executed. For a description of code options, see Section II.3. This section is broken down into an initial routine and a dynamic cycle. The initial routine prepares the code for execution; then the dynamic cycle integrates the time rate of change of the code variables over the time step to calculate their values. The dynamic cycle is repeated until execution is terminated either by the code or as predetermined by the user.

II.1 Initial Routine

The four parts of the initial routine are:

(i) read in the input data

(ii) write the input data to a file

(iii) initialize the variables

(iv) spray fire calculation

II.1.1 Input Data

The input data consists of titles and headings, control flags for the choice of options, geometries, initial conditions and material properties. User chosen options are described in Section II.3 Modeling Options. Input variables are described in Section III.1 LITFIRE and Input Data Files and examples of the input data files are given in Appendix D.

II.1.2 Print out the Input

The input is printed out to a file, which should be examined by the user to ensure that the input was properly entered into the input files. Misaligned input data is a common source of error that can easily be caught by examining the printed input.

II.1.3 Variable Initialization

The initialization section sets all time rates of change to zero for the first step. Some constants are defined and initial conditions are applied using the input data. In addition there is a short sub-section where the units of the input data are changed to the following to become consistent with the rest of the code:

BTU, pounds mass, feet, seconds

II.1.4 Spray Fire Calculations

The spray fire calculation simulates the effect of lithium reacting as a spray at the beginning of a spill. The amount reacted in the spray is user specified. A difficulty does arise in that the specific heats of the reaction products are calculated as functions of temperature. The spray fire calculation is not a spray fire model *per se* in that the lithium reacting in the spray is consumed instantaneously, adiabatically and stoichiometrically, with the equilibrium temperatures being determined by iteration.

II.2 Dynamic Cycle

The dynamic cycle in LITFIRE calculates the temperature, pressure and mass profiles of the reactor containment over the time of the run. Most of the dynamic cycle consists of calculating the thermal admittances between nodes, which are then used in conjunction with the nodal temperatures, to determine the time rates of change of the nodal temperatures. The heat flow to or from a node is determined by calculating the thermal resistances to conduction or convection between adjacent nodes and then adding on radiative heat transfer to or from nodes within a line of sight as shown below:

$$q_{12} = \frac{(T_1 - T_2)}{\sum_i \frac{1}{h_i A_i} + \sum_j \frac{\Delta x}{k_j A_j}} + \sum_k F_k \sigma_k A_k T_k^4 \tag{1}$$

Temperature rates of change are determined by:

$$(\frac{dT_1}{dt}) = \frac{1}{m_1 c_{p_1}} \sum_i q_{i1} \tag{2}$$

The time rates of change are then integrated over the time step by use of the fourth order Runge-Kutta Method or Simpson's Rule. The integrations are performed as follows:

$$T(t) = T(t_o) + \int_{t_o}^{t} \frac{dT}{dt} dt \tag{3}$$

In addition to the temperature calculations, the other calculations performed in the dynamic cycle are listed below in the order which they are performed:

(i) temperature dependent properties: heat capacities, gas fractions, radiative interchange factors and emissivities
(ii) perform an energy balance to determine the gas temperature if the steam in containment option is used
(iii) natural convection heat transfer coefficients
(iv) preliminary thermal admittances
(v) test for combustion/no combustion
(vi) temperature rates of change from heat flow
(vii) overpressure, leakage and aerosol sticking
(viii) perform integrals
(ix) check for terminating execution
(x) time step control
(xi) write the output to a file
(xii) display error pointers if necessary

Short descriptions of the subsections appear below.

II.2.1 Temperature Dependent Properties

Most properties are assumed to be constant with respect to temperature. However, the specific heats of some gases and combustion products, and most lithium properties, are calculated as a function of temperature. References for the heat capacity correlations and the derivation of the radiative interchange factors used in the code can be found in references 1 and 2.

II.2.2 Gas Node Temperature Determination

If the steam in containment option is being used, the presence of a condensible gas requires that a different method be used to determine the temperature of the gas than the integral method described in section II.2, as the latent heat of vaporization of the steam must be taken into account. In this option, the code performs an iterative energy balance by solving the equation:

$$\mathrm{TEMP} = U_v - M_a c_{v_a} T - M_s u_s \tag{4}$$

where U_v is the total internal energy of the gas, M_a is the mass of the non-condensible gas, c_{v_a} is the specific heat of the non-condensible gas, M_s is the mass of the steam and

u_s is the specific internal energy of the steam. Values of the temperature are guessed, and along with the specific volume of the steam (which is known), are used to determine the other properties of the steam. These values are then used to solve for the error, TEMP. When TEMP is a sufficiently small fraction of the total gas internal energy, the final guess is taken as the gas temperature. For further discussion, see reference 4.

II.2.3 Natural Convection Heat Transfer Coefficients

The heat transfer coefficients for convective heat transfer are determined from the Nusselt number Nu,

$$Nu = C(GrPr)^{\frac{1}{3}} \tag{5}$$

where

$$Nu = \frac{\mathrm{h}L}{k} \tag{6}$$

and the Grashof number Gr and the Prandtl number Pr are given by:

$$Gr = \frac{g\beta\Delta T L^3}{\nu^2} \tag{7}$$

$$Pr = \frac{\mu c_p}{k} \tag{8}$$

A separate constant C is used to calculate the Nusselt number for each convective heat transfer interface. These constants are listed in the glossary, Appendix B.

In the presence of steam, a condensible gas, the heat transfer coefficients used are the Uchida heat transfer coefficients determined empirically as a function of the ratio of the mass of water to the mass of non-condensible gases in the atmosphere. A more detailed description is given in reference 4.

II.2.4 Preliminary Thermal Admittances

This section generates time rates of change of temperature for nodes when they are independent of whether or not lithium combustion occurs. Most calculations for the user specified options fall into this category. These calculations are also performed after the combustion/no combustion test (Section II.2.5) when the two parallel branches of the code rejoin.

II.2.5 Test for Combustion or No Combustion

The temperature rates of change can differ greatly depending whether or not the lithium pool is actually combusting (reacting with the containment atmosphere). This section of the code determines whether or not the lithium is combusting and then sends the code either to the branch with a combustion zone node where the lithium reacts or the branch without a combustion zone node. The criteria used to determine whether or not the lithium is combusting are :

(i) liquid lithium must be available (between 180 and 1347°C.)

(ii) oxygen, nitrogen or water vapor must be available

(iii) if no oxygen or water vapor is present, the combustion zone temperature must be less than 1127°C.

II.2.6 Temperature Rates of Change

These calculations, as shown before, are based on summing the heat transfer from all thermally adjacent nodes. The heat transfer is determined from the basic relations for conductive, convective and radiative heat transfer:

$$\text{conduction: } \frac{dq}{dt} = kA\frac{dT}{dx} \tag{9}$$

$$\text{convection: } \frac{dq}{dt} = hA(T_1 - T_2) \tag{10}$$

$$\text{radiation: } \frac{dq}{dt} = \sigma A F_k(T_1^4 - T_2^4) \tag{11}$$

For a more detailed description, see references 1 and 2.

II.2.7 Overpressure, Leakage, and Aerosol Behavior

The masses of each cell gas component and the cell gas temperature are integrated in the integration section. From this, the cell gas pressure is determined and thus leakage from containment can be calculated. Aerosol adhesion to the containment walls is a user specified option and changes the concentration of aerosol combustion products in the containment atmosphere.

II.2.8 Integrals

All time rates of change are integrated over each time step to calculate the values of the quantities during the execution of the code. The form of the integrals is:

$$P = \text{INTGRL}\left(P_o, \frac{dP}{dt}\right) \tag{12}$$

where

P_o = the initial value of function P (mass, temperature or energy)

$\frac{dP}{dt}$ = the time dependent rate of change of P

INTGRL is an integration function that uses a fourth order Runge-Kutta Method or Simpson's Rule (user specified) to simultaneously solve all of the differential equations used in the code. For a more detailed description, see the listing of the code, Appendix F.

II.2.9 Termination Checks

The conditions that will terminate the code are:

(i) the lithium temperature reaches a value at which the lithium vaporizes (1347°C.) or solidifies (180°C.)

(ii) the primary cell gas temperature returns to ambient temperature with no overpressurization

(iii) the code reaches the user specified stopping point (TIME$\geq$TIMEF)

II.2.10 Time Step Control

Three criteria are used to determine the size of the time step used during each dynamic cycle. They are:

(i) the time step must be smaller than a user defined fraction of the inverse rates of change:

$$\text{DELT} < \text{RELERR T}\Big/\left(\frac{dT}{dt}\right)$$

(ii) the conduction heat transfer limit must be satisfied:

$$\frac{\alpha \Delta t}{(\Delta x)^2} < 0.3$$

(iii) The user imposed maximum and minimum time steps must be observed. The maximum time step is indicated by DELOUT and the minimum by DTMIN. DELOUT and DTMIN are read from the first input file.

II.2.11 Output Section

The output from LITFIRE is written into data files as the code is executed. Examples of the output files are shown in Appendix E, generated by the input files found in Appendix D.

II.2.12 Error Pointers

This section is not actually part of the dynamic cycle, although if an error during execution should occur, an error message would be written into file *out1.dat* and code execution would halt.

II.3 Modeling Options

The basic version of LITFIRE is capable of simulating a wide variety of spill conditions as indicated by the user in the first input data file. The containment volume, height, wall and floor areas, atmosphere, and material composition may be specified as well as the mass and surface area of the lithium spilled. Optional reaction geometries include a primary cell containing the lithium, surrounded by a larger secondary cell, and a partially insulated pan holding the lithium inside the basic primary cell. A concrete floor and wall for the containment are optional as is a liquid metal-concrete reaction routine. Also instead of elemental lithium, a lithium-lead eutectic may be selected for the spill with the composition chosen by the user. Finally an option to simulate a lithium spill in the presence of a steam-air atmosphere may be chosen.

In addition to the above options, several options involving the mitigation of lithium fires are available. These include:

(i) gas flooding

(ii) emergency space cooling

(iii) emergency floor liner cooling

(iv) aerosol removal

(v) gas injection

Each option is discussed below.

II.3.1 One Cell

The single cell model is the simplest version of LITFIRE that may be run. All other options are constructed as subroutines added on to the one cell version of the code. The nodes existing in the one cell version are shown in Figure II.3.1.1 There may be up to twenty concrete wall or floor nodes of any thickness. As stated earlier all material properties may be chosen by the user as may the composition of the containment atmosphere. Lithium or lithium-lead may react with any mixture of oxygen, nitrogen or water vapor and lithium only may react with carbon dioxide in the absence of oxygen. Any inert gas may also be included in the cell atmosphere. Lastly, any containment or spill geometry may be chosen as stated above.

The heat transfer correlations are fixed by the code so that accurate results may be obtained, although some modification is possible from the input data as shown in section II.2.3. The heat transfer pathways are also fixed by the code and are shown in Figure II.3.1.2. The heat transfer mechanisms and their applications to the code are discussed in references 1,2 and 3.

II.3.2 Two Cell

The two cell option was developed to model the effects of a fire inside a tokamak fusion reactor and to determine its effects on the structural integrity of the the torus. In this option a separate secondary cell with its own material composition, atmosphere and geometry exists surrounding the primary cell. (See Figure II.3.2.1) The provision for a crack between the primary and secondary cells, allowing the exchange of cell gases also exists. The code follows the composition, pressure and temperature of both cell gases during the run. The heat and mass flow paths in two cell LITFIRE are shown in Figures II.3.2.2 and II.3.2.3. High velocity gas flows, as would be encountered in the event of a breach in a vacuum torus have been successfully modeled by LITFIRE. (See reference 3)

II.3.3 Pan

Figure II.3.3 shows the pan option available in LITFIRE. This option may be used with either the one or two cell option, but not with concrete reaction. This option was created to model the lithium fire experiments performed at HEDL. The pan dimensions and composition are user defined. It contains two separate insulation nodes which transfer no heat to their surroundings.

II.3.4 Concrete Reaction

The liquid metal-concrete reaction subroutine allows for the reaction of lithium with the concrete floor under the liner. This includes lithium reactions with water driven from the concrete and certain components of the concrete itself. For further discussion, see reference 2.

II.3.5 Lithium-Lead Combustion

This option allows for the substitution of a lithium-lead eutectic in the place of elemental lithium as the source of the fire. All LITFIRE options are compatible with the lithium-lead combustion option except the initial spray fire calculation (Section II.1.4). In addition, this option allows the user to chose either a layered or turbulent pool reaction, allowing for optimistic or pessimistic results, depending on the user's view of lithium-lead fires. Reference 3 discusses the lithium-lead option in greater detail.

II.3.6 Steam-Air Atmosphere

The steam-air atmosphere option allows for the reaction of lithium or lithium-lead with a mixture of steam and other non-condensible gases. It also includes modifications of the convective heat transfer coefficients to account for condensation and the provision for steam condensation into a water pool node above the cell floor liner. Gas temperature is determined by an iterative energy balance routine (as shown in section II.2.2) to account for the presence of the condensible vapor, rather than in the usual integral method described earlier. The steam in the atmosphere may be present as humidity or may be injected into the primary or secondary cell atmosphere, with the time, mass flow rate and enthalpy of the steam injected selected by the user. The development of the steam-air atmosphere option is discussed in reference 4.

II.3.7 Mitigation Options

The following is a list of options available to evaluate the effectiveness of certain techniques used to attempt to mitigate the consequences of a lithium fire:

(i) gas flooding
(ii) emergency space cooling
(iii) emergency floor liner cooling
(iv) aerosol removal
(v) gas injection

Each option allows the user to select additional heat removal mechanisms as desired. These options are discussed further in reference 1.

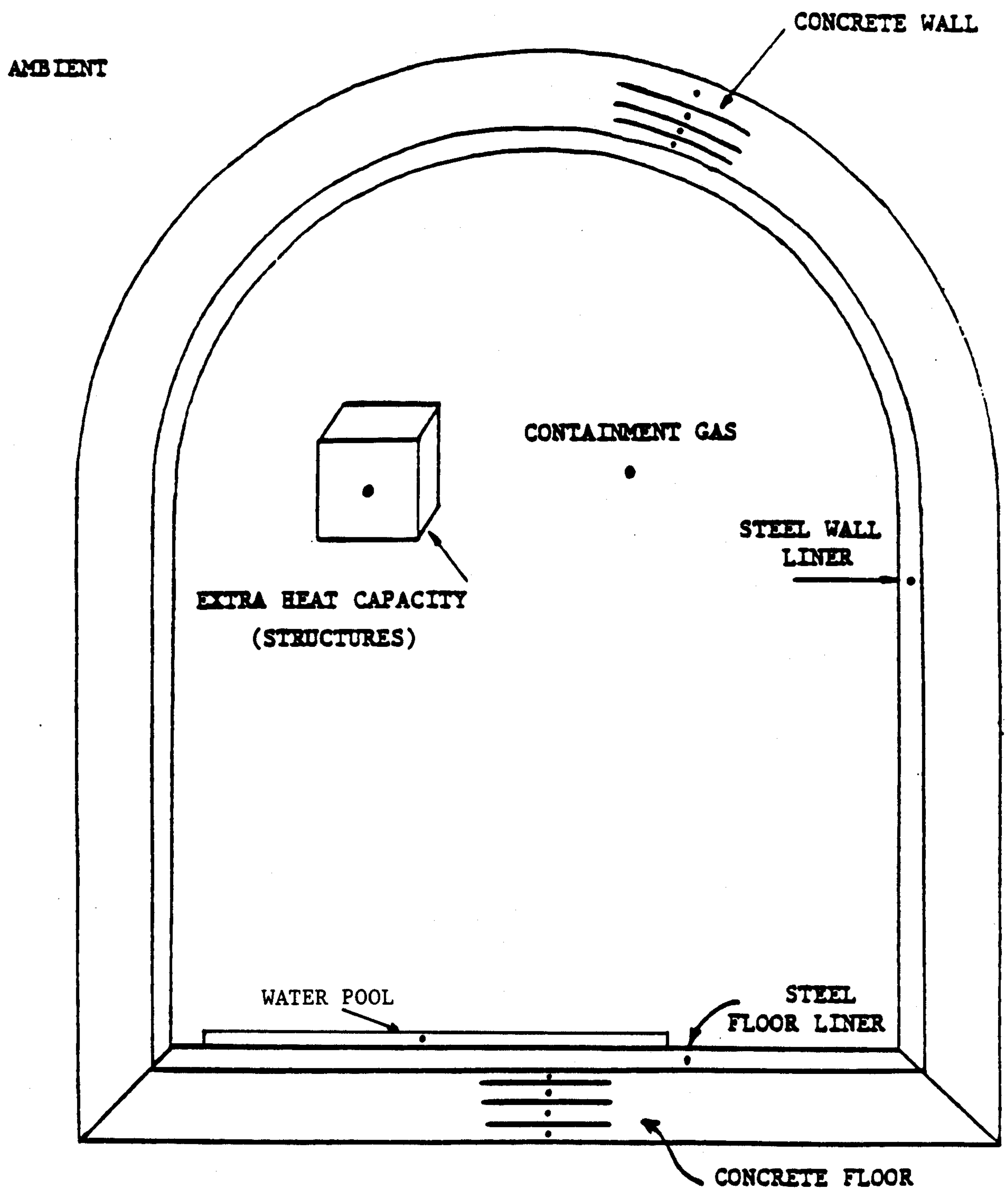
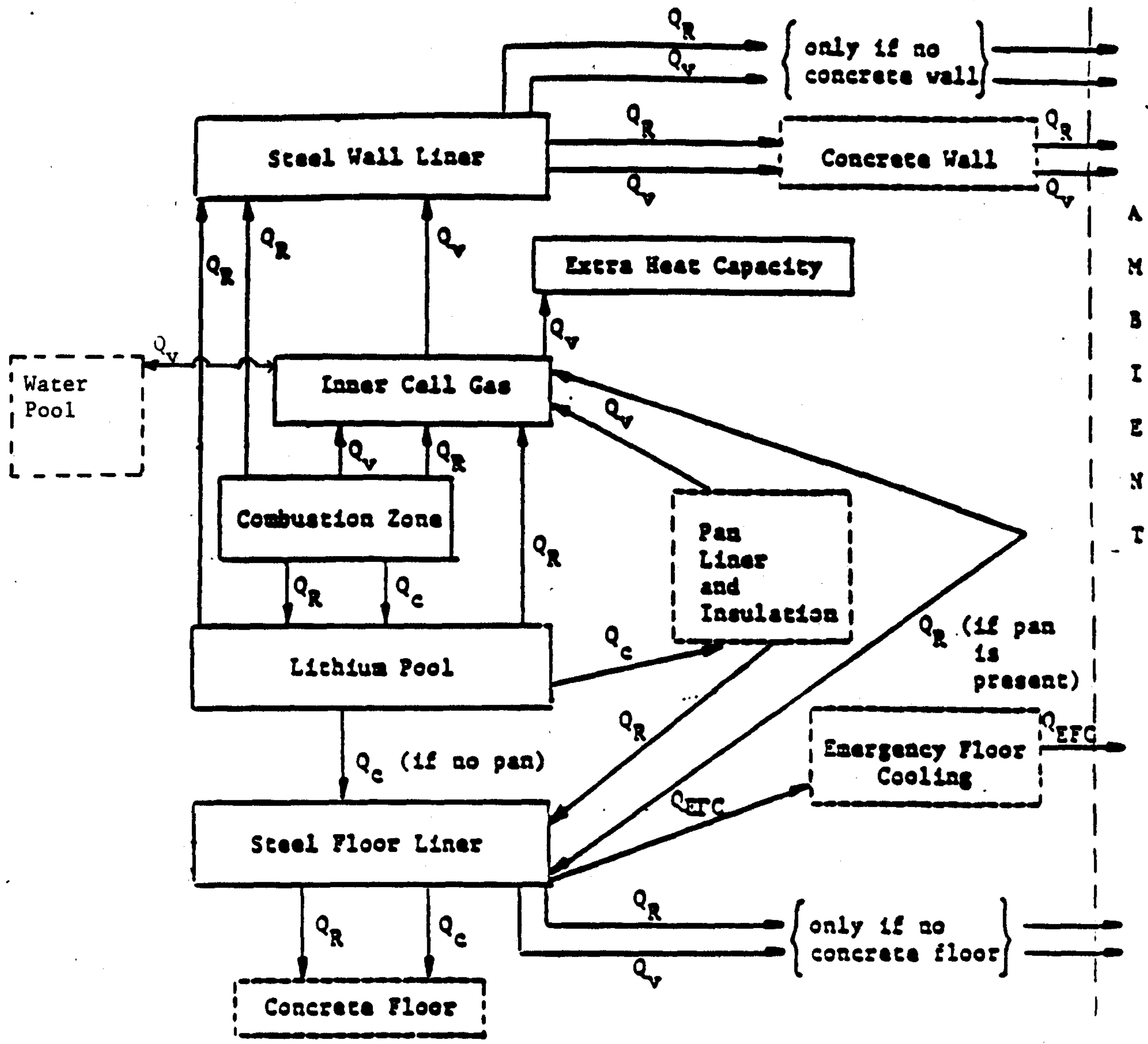


Figure II.3.1.1 One-Cell Geometry



dashed lines indicate optional node

Q_R = radiative heat transfer

Q_v = convective heat transfer or water condensation

Q_c = conductive heat transfer

Figure II.3.1.2 Energy flow in single cell LITFIRE

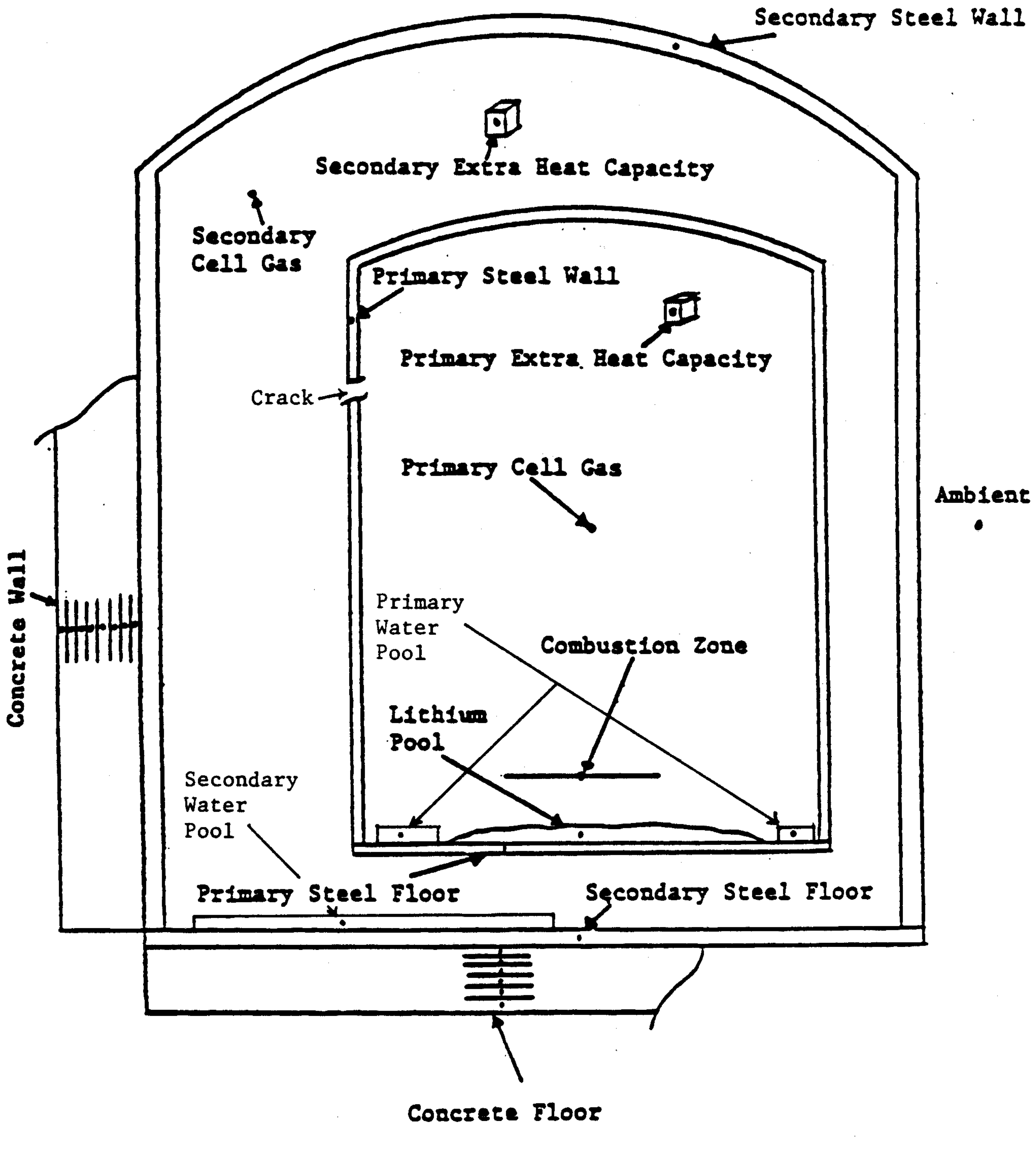


Figure II.3.2.1 Two-Cell Geometry

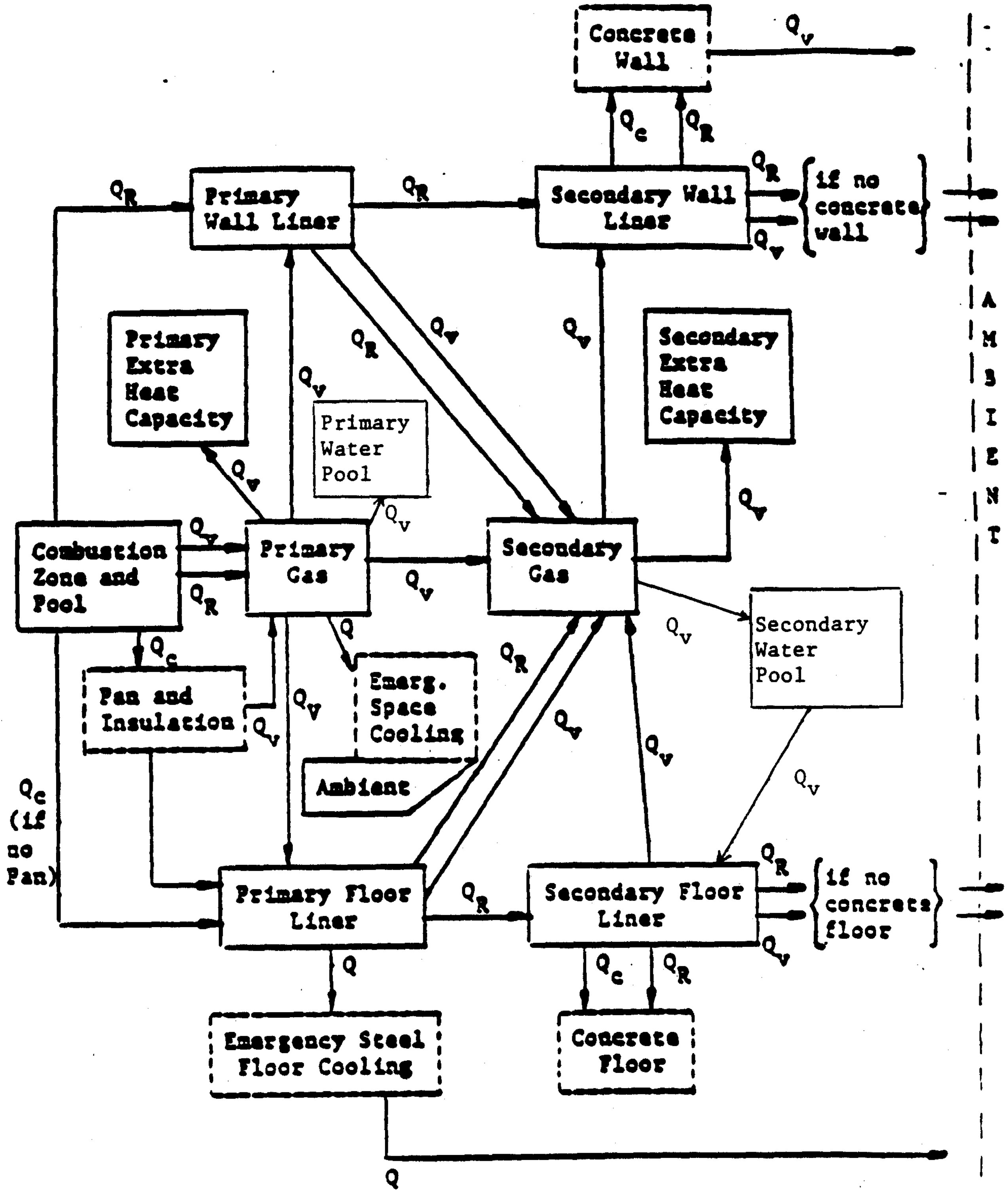
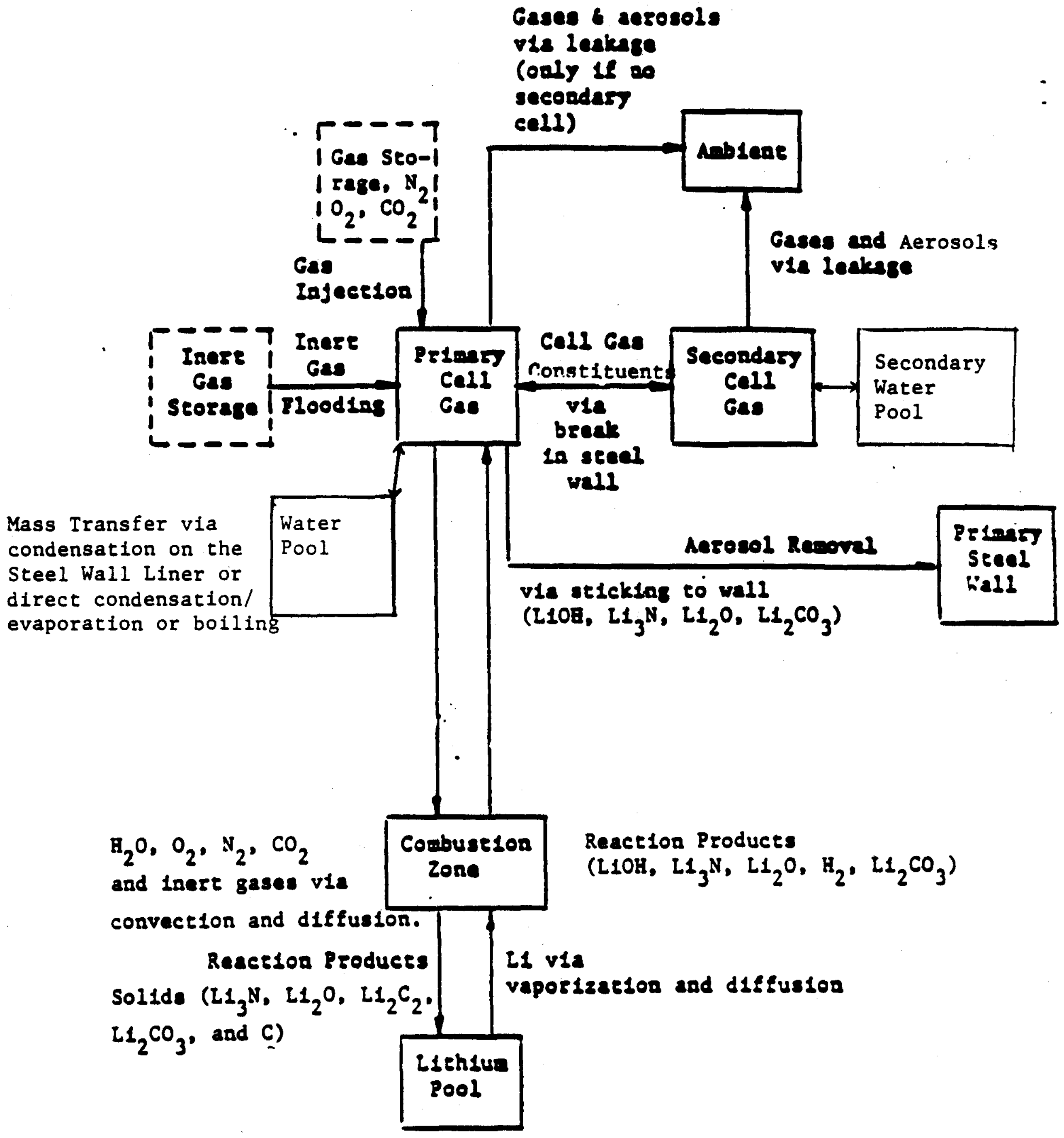


Figure II.3.2.2 Energy flow in two-cell LITFIRE



Gases & aerosols via leakage (only if no secondary cell)
Ambient
Gas Storage, N_2 O_2, CO_2
Gas Injection
Gases and Aerosols via leakage
Inert Gas Storage
Inert Gas Flooding
Primary Cell Gas
Cell Gas Constituents via break in steel wall
Secondary Cell Gas
Secondary Water Pool
Mass Transfer via condensation on the Steel Wall Liner or direct condensation/ evaporation or boiling
Water Pool
Aerosol Removal
via sticking to wall ($LiOH$, Li_3N, Li_2O, Li_2CO_3)
Primary Steel Wall
H_2O, O_2, N_2, CO_2 and inert gases via convection and diffusion.
Combustion Zone
Reaction Products ($LiOH$, Li_3N, Li_2O, H_2, Li_2CO_3)
Reaction Products Solids (Li_3N, Li_2O, Li_2C_2, Li_2CO_3, and C)
Li via vaporization and diffusion
Lithium Pool

Figure II.3.2.3 Mass flows in LITFIRE

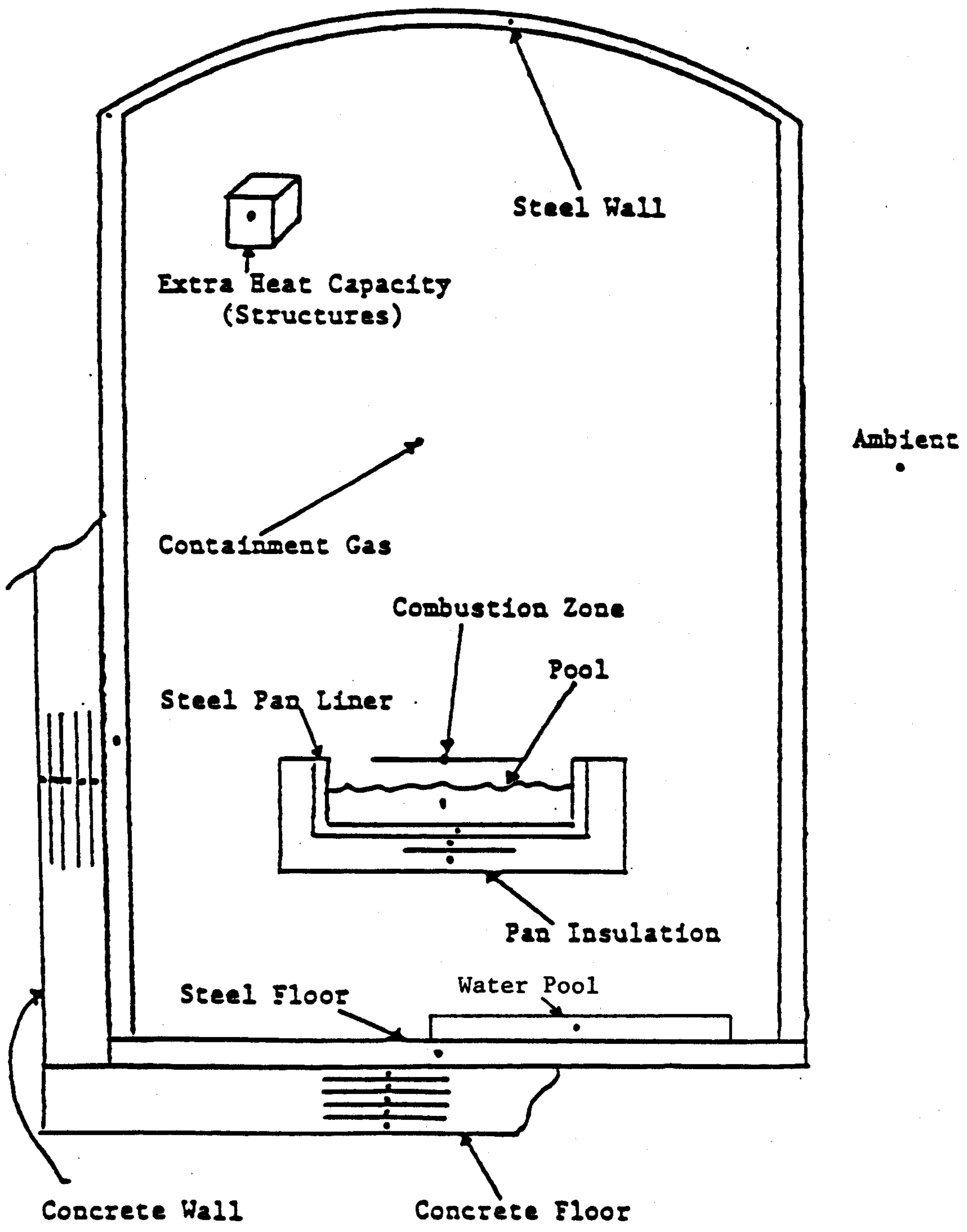


Figure II.3.3 One-Cell with Pan Geometry

III. Execution of LITFIRE

The execution of the LITFIRE code requires certain system commands depending upon the location at which it is being run. The specific commands for compiling, loading (linking), and running the code will be discussed in this section in the following order:

(i) Source code and sample input files

(ii) PFCVAX

(iii) MFE Crays

III.1 Obtaining Copies of LITFIRE and Input Data Files

There are currently two locations at which a copy of the source code of LITFIRE can be found: the PFCVAX at MIT, and the MFE FILEM disk storage.

To use the PFCVAX copy, the user must first obtain a PFCVAX account. This may be done by contacting the Plasma Fusion Center at MIT. Introductory information may be found by using the HELP command, which will define most of the commands used on the PFCVAX.

To get a copy of LITFIRE for personal use, the following commands must be used:

copy [barnett.litfire]litfire.for [username].**

This will copy the code *litfire.for* into the user's main directory.

To get a copy of the sample input files, the *copy* command must also be used, substituting the following filenames for *litfire.for* :

head.dat

uwmak.w

uwmak.x

uwmak.y

uwmak.z

steamop.

After obtaining copies of the input files, they may be viewed by using the *type* command or edited by using any line or text editor. (e.g. EMACS or EDT).

To obtain a copy of the LITFIRE source code on one of the MFECC machines, the user must be logged onto a machine and then use the *filem* command to obtain a personal copy:

filem read .15467 litfire

This will copy litfire into the user's personal directory on that particular machine.

To obtain copies of the input files, the command

read .15467 filename

must be used for each input file (the names are the same for MFE and the PFCVAX) while still in *filem.*

Once copies of LITFIRE and the sample input files have been obtained, they may be moved to a different machine '*n*' by using the *netout* command as follows:
netout filename site=nma

III.2 Organization of LITFIRE for Execution

One of the most important steps in the execution of LITFIRE is the preparation of the input data files. Most errors in code execution occur due to mistakes in the input data files. The input data files and the options they control are listed below. As stated in section II.3, some options are incompatible with each other.

uwmak.w:	one cell
uwmak.x:	two cell
uwmak.y:	pan, concrete reaction and lithium lead combustion
uwmak.z:	gas flooding, emergency space cooling, emergency floor liner cooling, aerosol removal and gas injection
steamop.:	steam injection with steam-air atmosphere

III.2.1 One Cell Option

The one cell option is the simplest version of LITFIRE. The first input file must exist to run any other code options. The order in which the input variables must appear in the file *uwmak.w* is shown below (British units are given in the Glossary, SI units may be used for all input data files by setting the input variable IFLAGISI = 1):

line	variables
1	IFLAGW,IFLAGF,IFLAGP,IFLAG2,IFLAGS,IFLAGC, IFLAGU,IFLAGB,IBLOW,IESC,ISFLC,ISWICH, IAROSL,IFLAGD,IFLAGISI,IFLAGCO,IFLAGST
2	NL,NL1
3	L(1) to L(NL)
4	L1(1) to L1(NL)
5	VP,CHP,CPAP,XMOLA
6	TEHCZP,XMEHCP,AEHCP,CPEHCP,HINECP
7	THWC,THFC,GAP,KGAP,KLEAK
8	ESTLWP,CPSWP,KSTLWP,RHSWP,AWP,THWP
9	ESTLFP,CPSFP,KSTLFP,RHSFP,AFP,THFP
10	EMLI,CPLI,AKLI,RHLI
11	EMCONC,CPCON,KCON,RHCON
12	RHOLIO,RHOLIN,RHOLIH,EMGPF,EMCZ,TAUCZ
13	QCO1,QCO2,QCN,QCW
14	RCMBO1,RCMBO2,RCMBN,RCMBW,RCMBH2
15	TMELT,TVAP,QVAP,PERCEN
16	CONF1,CONF2,C2FAC
17	HIN,HINGSP,HINGSS,HINPS,HINSAM,HINFAN
18	HINGFS,HINFSG
19	ASLI,SPILL,SPRAY,FRA,RA
20	TCZI,TGPZER,TSPZER,TSFPI,TA,TLII
21	PAPZER,WO2P,HUM,WAP,WCP
22	IMETH,DTMIN,TIMEF,RELERR,DELOUT,OUTPUT

The formats for the input lines are:

line	formats
1	(1x,14(i1,1x))
2	(i4,i4)
3,4	(10f5.3)
5	(f12.2,5f12.4)
6–21	(6f12.4)
22	(i4,5f12.4)

An example of this data file (British units) is *uwmak.w* in Appendix D.

III.2.1.1 PFCVAX

The statement:

open(unit=2,file='uwmak.w',status='old')

must be included in the code for the code to execute, assuming that the input data file is named *uwmak.w*. A similar OPEN statement must be included for each input data file used by the code. These statements are currently located after the "common" blocks at the beginning of the code.

III.2.1.2 MFE Crays

For these machines the statement:

call link("unit1=filename,read1//")

must be used for each input data file. These statements are also currently located after the "common" blocks at the beginning of the code.

III.2.2 Two Cell Option

If the two cell option is chosen, the following are the variable listing and formats for the second data file (units are given in the glossary as for the first input file):

line	variables
1	VS,CHS,PASZER,TGSZER,TSSZER,TFSZER
2	CRACK,HUM2,WO2S,WAS,CPAS,WCO2S
3	TEHCZS,XMEHCS,AEHCS,CPEHS,HINECS
4	ESTLWS,CPSWS,KSTLWS,RHSWS,AWS,THWS
5	ESTLFS,CPSFS,KSTLFS,RHSFS,AFS,THFS
6	TSWICH

The format for all lines is: (6f12.4)

III.2.2.1 PFCVAX

If this option is used, the statement:

open(unit=3,file='filen.ame',status='old') must be included in LITFIRE in the same place as the OPEN statement for the first input data file. An example of this data file (British units) is given in Appendix D.

III.2.2.2 MFE Crays

For these machines, the statement:

call link("unit3=filename,read3//")

must be used in the same location as the CALL LINK statement for the first input data file.

III.2.3 Pan, Concrete Reaction and Lithium-Lead Combustion

The following is a variable listing for each of the above options (British units are given in the Glossary):

Pan Option

line	variables
1	KPAN, RHPAN,CPPAN,RHINS,CPINS,EMINS
2	TPANZO,APAN,BREDTH,AINS,HINGF
3	THKPAN,THKIN1,THKIN2

Concrete Reaction Option

line	variables
1	ZZDIN,QCCONC,CRACON,XMH2OI,TCIGNI,RCMBC

- the pan option cannot be chosen with concrete reaction

Lithium-Lead Option

line	variables
2(1)	CPLEAD,KLEAD,RHLEAD,ALLOYI,QDISS

- Lithium-lead data is entered on line 1 if the concrete reaction option is not chosen. The pan option cannot be chosen with lithium-lead combustion.

The format for all lines is: (6f12.4)

III.2.3.1 PFCVAX

The statement:

open(unit=4,file='filen.ame',status='old')

must be used. An example of this file (British units) is given in Appendix D.

III.2.3.2 MFE Crays

For these machines, the statement:

call link("unit4=filename,read4//")

must be used in the same location as the CALL LINK statement for the first input data file.

III.2.4 Gas Flooding, Emergency Space Cooling, Emergency Floor Liner Cooling and Gas Injection

Each of these options is included in one input data file. The following is a variable listing for each of the options. Only the variable lines of the options used need be entered, although the lines used must be in the order shown below (British units are given in the Glossary):

line		variables
1	(gas flooding)	WO2B,WWAB,WN2B,XMOLAB,CPAB,TBLOW
2		BLOWV,EXHSTV,TBLIN,TBLOUT,WAB
3	(emerg. space cooling)	ESCR,ESCTIN,ESCEND
4	(emerg. floor cooling)	SFLCR,SFLTIN,SFLEND
5	(aerosol removal)	BETA
6	(gas injection)	TONE,TTWO,TTHREE,DP1,DP2,DP3,FCT1,FCT2,FCT3

The format for lines 1–5 is: (6f12.4). For the gas injection option (line 6) it is: (3f10.2,6f8.4).

III.2.4.1 PFCVAX

The statement:

open(unit=5,file='filen.ame',status='old')

must be used. An example of this file (British units) is given in Appendix D.

III.2.4.2 MFE Crays

For these machines, the statement:

call link("unit5=filename,read5//")

must be used in the same location as the CALL LINK statement for the first input data file.

III.2.5 Steam Injection to Containment

The following is a variable listing for the steam injection to containment available in the steam-air atmosphere option (British units are given in the Glossary):

line	variables
1	STMIN,STMOUT,MINJR,HINJ
2	STMIN2,STOUT2,MINJR2,HINJ2

The format for all lines is: (6f12.4)

III.2.5.1 PFCVAX

The statement:

open(unit=6,file='filen.ame',status='old')

must be used. An example of this file (British units) is given in Appendix D.

III.2.5.2 MFE Crays

For these machines, the statement:

call link("unit6=filename,read6//")

must be used in the same location as the CALL LINK statement for the first input data file.

III.3 PFCVAX Execution

To execute LITFIRE on the PFCVAX, the user must ensure that all necessary OPEN statements are included for the data files necessary to run the code as indicated in section III.2. The input data file *head.dat* must always be included as it provides headings for the output data files that LITFIRE will create. The statement necessary to include *head.dat* is:

open(unit=1,file='head.dat',status='old')

It should be noted that all input data files must be in the same directory as LITFIRE in order to execute the code.

OPEN statements are also needed for the output files created by LITFIRE. These statements are placed right after the OPEN statements for the input data files and take the form of:

open(unit=10,file='out1.dat',status='new')

There are ten output data files named *out1.dat* through *out10.dat*, and an OPEN statement is needed for each one.

Optional output files may be created to allow graphs to be made of code variables vs. time. These files may be used in conjunction with the graphics routines available on the PFCVAX to create the actual graphs. The optional files require OPEN statements like the ones for *out1.dat* through *out10.dat*, but use the filename *for0nn.dat*, where *nn* is any number from 1 to 99. Steps must be added to the code in the output section to write values of TIME and the variable desired to the file. Some optional output file OPEN and WRITE statements which have been commented out may be found in the appropriate sections of the code. The output files created may be used with the McCool graphics routine available on the PFCVAX to create the graphs.

There are three separate steps involved in the execution of LITFIRE: compiling, linking and running the code. To compile LITFIRE, the command:

for litfire.for

must be used and will create the file *litfire.obj.* To link LITFIRE, the command:

link litfire.obj

must be used and will create the file *litfire.exe.* To run LITFIRE, the command:

run litfire.exe

must be used. If the input data files were entered properly, the code should run until terminated by a user defined time limit (i.e., TIME=TIMEF) or by the code itself. (e.g., 'the temperature of the lithium pool has reached the melting point'). If the code stops due to an error, an error message will be given. Refer to Appendix C—Troubleshooting.

III.4 MFE Crays Execution

To execute LITFIRE on the Crays, the necessary CALL LINK statements must be present for all of the input data files as indicated in section III.2. The input data file *head.dat* must be included to provide headings for the output data files that LITFIRE will create. The statement needed to include *head.dat* is:

call link("unit1=head.dat,read1//")

CALL LINK statements are needed for the ten output files as well. Those statements take the form of:

call link("unit10=(out1,text,create),print1//")

for each of the ten output files *out1* through *out10.* They should be placed directly after the CALL LINK statements for the input data files.

To compile, load and run LITFIRE on the Crays, the following commands should be used:

$rcft\ i = litfire, x = xlitfire\ /\ t\ p$

$xlitfire\ /\ t\ p$

where t is the total CPU time allowed in minutes and p sets the priority of the run. One minute and a priority of 2 should be sufficient to run *xlitfire*.

III.5 Sample Input and Output Files

To check whether LITFIRE has been properly executed, a set of sample input and output files are given in Appendices D and E. Execution of the code using the sample input files on the PFCVAX or the Crays will produce approximately the same results as given in the sample output files.
Using the input data files:

head.dat, uwmak.w, uwmak.x, uwmak.y, uwmak.z, steamop.

the following output files should be generated:

out1.dat, out2.dat, out3.dat, out4.dat, out5.dat, out6.dat, out7.dat, out8.dat out9.dat, out10.dat.

It should be noted that only the input files *head.dat, uwmak.w* and *uwmak.x* were used for the examples in the Appendices.

APPENDICES

Appendix A

Nomenclature

$A_{i,j,k}$	heat transfer surface areas
c_p	specific heat
c_{v_a}	specific heat at constant volume
F_k	radiative view factor, including emissivity
g	acceleration due to gravity
Gr	Grashof number
h	convective heat transfer coefficient
k, k_j	thermal conductivity
L	characteristic length
M, m	mass
Nu	Nusselt number
Pr	Prandtl number
dq/dt	heat flow rate
t	time
T	temperature
u	specific internal energy
U_v	total internal energy
x	linear distance
α	thermal diffusivity
β	coefficient of volumetric expansion
μ	fluid viscosity
ν	kinematic viscosity
ρ	gas density
σ, σ_k	Stefan-Boltzmann constant

Appendix B

Variable Listing

AA	Exponent used in expression for GRPR or GRPRF
ACTVTY	Calculates activity of lithium in lithium-lead eutectic
AEHCP	Surface area of primary extraneous heat capacity (ft^2)
AEHCS	Surface area of secondary extraneous heat capacity (ft^2)
AFP	Surface area of primary floor liner, must be equal to or greater than the area of the lithium spill ASLI. (ft^2)
AFS	Surface area of the secondary floor liner (ft^2)
AHT	Area of the lithium pool-pan interface (ft^2)
AINS	Outside exposed area of insulating layer on the pan (ft^2)
AIRFAC	Density weighting factor for calculating steam-air mixture properties
AK1	Product of thermal conductivity and Prandtl number of the primary gas (BTU/sec-ft deg. F.) See associated film temperature T1.
AKEXX	Function used to calculate heat transfer coefficients
AKLEAD	Thermal conductivity of lead (input as BTU/hr-ft deg. F.)
AKLI	Thermal conductivity of lithium (input as BTU/hr-ft deg. F.)
AKLIX	Thermal conductivity of lithium used during LC-2 lithium transfer simulation (BTU/sec-ft-deg. F.)
AKVAP	Product of thermal conductivity and Prandtl number of water vapor at the lithium pool surface (BTU/sec-ft deg. F.)
ALLOYI	Initial atom percent of lithium in lithium-lead
ALPHA	Used to determine whether or not LILP should be fixed at an amount equal to AKLI/(RHLI*CPLI)
ALPHA2	Used in determining PYU. Also tests conduction limit on time step for the pan or floor liner.
AMIN1	FORTRAN function that determines the minimum of the arguments used
APAN	Pan external heat transfer area (ft^2)
ASLI	Surface area of the lithium spilled (ft^2)
ASURF	Surface area of the liquid water pool (ft^2)
AWP	Surface area of the primary wall liner (ft^2)
AWS	Surface area of the secondary wall liner (ft^2)
B	Used in calculating the thermal resistance between the wall liner, the gap and the wall concrete
BB	Analogous to B, but for the floor liner, gap and concrete
B1	Coefficient of volumetric expansion for gas (β) (1/deg. F.) See associated film temperature T1
BETA	Inverse sticking coefficient for particles impinging on the wall (sec.)
BETAB	Volumetric expansion coefficient β for the water pool-gas boundary (1/deg. F.)
BETAF	Volumetric expansion coefficient β for the water pool (1/deg. F.)
BIL	Fractional change between BILGE and DELT, used in determining minimum time step
BILGE	Equal to the minimum value of DT1,DT2,DT3,DT4 or DT5, used in

	calculating the time step length (sec.)
BLIN	Time after spill at which inert gas flooding and exhaust begins (sec.)
BLOUT	Time after spill at which inert gas flooding and exhaust ends (sec.)
BLOWR	Inert gas input rate (lbm/sec)
BLOWV	Inert gas volumetric input rate (ft^3/sec)
BREAKS	Outer cell gas temperature rate of change due to gas flow between cells and leakage. (deg. R./sec)
BREDTH	Perimeter of the pan (ft)
Cxxx	'C' is used to indicate a thermal admittance between nodes (i.e., the inverse of the product of effective thermal resistance between the nodes and the heat capacity of one of them (hA/mc_p) (sec^{-1})
C1	Primary gas to primary wall liner in gas
C2	Pan to primary gas in gas
C3	Wall liner to concrete in concrete
C4(i)	Concrete wall node i to node i+1 in concrete
C5	Concrete wall to ambient in concrete
C6	Primary gas to primary wall in wall liner
C7	Wall liner to concrete in wall liner
C8	Floor liner to concrete floor in floor liner
C9	Floor liner to concrete floor in concrete
C10(i)	Concrete floor node i to node i+1 in concrete
C11	Wall liner to ambient (no concrete option) in wall liner
C12	Floor liner to ambient (no concrete option) in floor liner
C13	Pan to primary gas in pan
C14	Secondary floor liner to secondary gas in floor liner
C15	Secondary floor liner to secondary gas in gas
C16	Primary floor liner to primary gas in floor liner
C17	Primary floor liner to primary gas in gas
C18	Primary floor liner to secondary gas in floor liner
C19	Primary floor liner to secondary gas in gas
C20	Primary wall liner to secondary gas in wall liner
C21	Secondary wall liner to secondary gas in wall liner
C22	Primary wall liner to secondary gas in gas
C23	Secondary wall liner to secondary gas in gas
C2FAC	Fraction of Li-CO_2 reaction which produces Li_2C_2
CA	Coefficient in expression for GRPR or GRPRF
CCZ	Amount of heat being developed in the combustion zone (BTU/sec)
'C'CZP	Lithium pool to combustion zone in pool
CD	Coefficient of discharge between the two cells (near unity)
'C'EHCGP	Primary extraneous heat capacity to primary gas in gas
'C'EHCGS	Secondary extraneous heat capacity to secondary gas in gas
'C'GCZ	Combustion zone to primary gas in combustion zone
'C'GLI	Lithium Pool to primary gas (no combustion) in pool
'C'GPEHC	Primary gas to primary extraneous heat capacity in heat capacity

'C'GSEHC	Secondary gas to secondary extraneous heat capacity in heat capacity
CHP	Primary cell height (ft)
CHS	Secondary cell height (ft)
'C'IN1PN	Pan to inner insulation in insulation
'C'IN12	Inner pan insulation to outer pan insulation in inner insulation
'C'IN21	Inner pan insulation to outer pan insulation in outer insulation
'C'LIG	Lithium pool to primary gas (no combustion) in gas
'C'LIPAN	Lithium pool to pan in pool (suspended pan option)
'C'LIST	Lithium pool to primary floor liner in pool
CMBR	Total combustion rate (lb Li/sec-ft^2)
CMBRC2	Combustion rate for the carbon reaction (lb Li/sec-ft^2)
CMBRCO	Combustion rate for the lithium-carbonate producing reaction (lb Li/sec-ft^2)
CMBRH	Total combustion rate (lb Li/hr-ft^2)
CMBRHH	CMBRH in g Li/min-cm^2
CMBRHI	Initial combustion rate (lb Li/hr-ft^2)
CMBRN	Combustion rate for the nitrogen reaction (lb Li/sec-ft^2)
CMBRNH	CMBRN in g Li/min-cm^2
CMBRO	Total of CMBRO1 and CMBRO2 (lb Li/sec-ft^2)
CMBRO1	Combustion rate for the oxygen reaction (lb Li/sec-ft^2)
CMBRO2	Combustion rate for the Li-CO_2 reaction producing lithium-oxide (lb Li/sec-ft^2)
CMBROH	CMBRO in g Li/min-cm^2
CMBRW	Combustion rate for the water vapor reaction (lb Li/sec-ft^2)
CMBRWH	CMBRW in g Li/min-cm^2
CMRC2H	CMBRC2 in g Li/min-cm^2
CMRCOH	CMBRO2 in g Li/min-cm^2
CO2	A subroutine which sets up the pure CO_2 atmosphere
CO2LFS	Carbon dioxide left after spray fire (lb)
CONDR	Condensation mass flow rate (lb/sec)
CONF1	Fraction of Li-CO_2 reaction which produces Li_2O
CONF2	Fraction of Li-CO_2 reaction which produces Li_2CO_3
CPxxx	Gaseous specific heats are all specific heats at constant volume
CPA	Primary non-condensible gas specific heat (BTU/lb deg. F.)
CPA2	Secondary non-condensible gas specific heat (BTU/lb deg. F.)
CPAB	Flooding gas specific heat (BTU/lb deg. F.)
'C'PANLI	Lithium pool to pan in pan
CPAP	Specific heat of primary cell inert gas (BTU/lb deg. F.)
CPAS	Specific heat of secondary cell inert gas (BTU/lb deg. F.)
CPB	Specific heat of the water pool-gas boundary (BTU/lb deg. F.)
CPCARP	Specific heat of carbon in the primary gas (BTU/lb deg. F.)
CPCO2P	Specific heat of carbon dioxide in the primary gas (BTU/lb deg. F.)
CPCO2S	Specific heat of carbon dioxide in the secondary gas (BTU/lb deg. F.)
CPCON	Heat capacity of floor and wall concrete (BTU/lb deg. F.)

'C'PCZ	Lithium pool to combustion zone in combustion zone
CPEHCP	Specific heat of primary extraneous heat capacity (BTU/lb deg. F.)
CPEHCS	Specific heat of secondary extraneous heat capacity (BTU/lb deg. F.)
CPFAC	Used in calculating CPLI (CPFAC=.004938^{TLI}-6.20741)
CPH2	Specific heat of hydrogen gas (2.48 BTU/lb deg. F.)
CPINS	Specific heat of insulation (BTU/lb deg. F.)
CPLC2P	Specific heat of lithium carbide in the primary gas (BTU/lb deg. F.)
CPLC3P	Specific heat of Li_2CO_3 in the primary gas (BTU/lb deg. F.)
CPLC3S	Specific heat of Li_2CO_3 in the secondary gas (BTU/lb deg. F.)
CPLEAD	Specific heat of pure lead (BTU/lb deg. F.)
CPLI	Specific heat of lithium (BTU/lb deg. F.)
CPLIX	Specific heat of lithium (used during LC-2 lithium transfer simulation (BTU/lb deg. F.)
CPLIH	Specific heat of lithium hydroxide (0.67 BTU/lb deg. F.)
CPLIN	Specific heat of lithium nitride (BTU/lb deg. F.)
CPLINP	Specific heat of lithium nitride in primary (BTU/lb deg. F.)
CPLINS	Specific heat of lithium nitride in secondary (BTU/lb deg. F.)
CPLIO	Specific heat of lithium oxide (BTU/lb deg. F.)
CPLIOH	Molar specific heat of lithium hydroxide (BTU/lb-mol deg. F.)
CPLIOP	Specific heat of lithium oxide in primary (BTU/lb deg. F.)
CPLIOS	Specific heat of lithium oxide in secondary (BTU/lb deg. F.)
CPLV	Specific heat of water for secondary floor liner-secondary water pool heat transfer (at TAVE) (BTU/deg. F.)
CPMCOP	Heat capacity of carbon dioxide in primary (BTU/deg. F.)
CPMCOS	Heat capacity of carbon dioxide in secondary (BTU/deg. F.)
CPMCZ	Effective heat capacity of combustion zone (BTU/deg. F.)
CPMH2	Heat capacity of hydrogen in containment (BTU/deg. F.)
CPMLCP	Heat capacity of lithium carbonate in primary (BTU/deg. F.)
CPMLCS	Heat capacity of lithium carbonate in secondary (BTU/deg. F.)
CPMLOP	Heat capacity of lithium oxide in primary (BTU/deg. F.)
CPMLOS	Heat capacity of lithium oxide in secondary (BTU/deg. F.)
CPMNIP	Heat capacity of nitrogen in primary (BTU/deg. F.)
CPMNIS	Heat capacity of nitrogen in secondary (BTU/deg. F.)
CPMOXP	Heat capacity of oxygen in primary (BTU/deg. F.)
CPMOXS	Heat capacity of oxygen in secondary (BTU/deg. F.)
'C'PNIN1	Pan to inner insulation in pan
CPN2P	Specific heat of nitrogen gas in primary (BTU/lb deg. F.)
CPN2S	Specific heat of nitrogen gas in secondary (BTU/lb deg. F.)
CPSFP	Specific heat of primary floor liner (BTU/lb deg. F.)
CPSFS	Specific heat of secondary floor liner (BTU/lb deg. F.)
CPSWP	Specific heat of primary wall liner (BTU/lb deg. F.)
CPSWS	Specific heat of secondary wall liner (BTU/lb deg. F.)
CPWV	Specific heat of water vapor in primary (BTU/lb deg. F.)
CRACON	Area of concrete exposed to lithium in concrete combustion model (ft^2)

CRACK	Area of the orifice between the two cells (square inches)
'C'SBLI	Lithium pool to primary floor liner in floor liner
DAB	Diffusion coefficient for air and water (ft^2/sec)
DELMP	Fractional exchange rate of primary gas used in determining the minimum time step (sec)
DELMS	Fractional exchange rate of secondary gas used in determining the minimum time step (sec)
DELOUT	User defined maximum time step length (sec)
DELT	Time step length (sec)
DFILM	Lithium vapor film thickness (ft)
DFLIPB	Diffusion coefficient for lithium through lead (ft^2/sec)
DIFF	Gas mass diffusion coefficient to the combustion zone (ft^2/sec)
DIFFLI	Lithium diffusion coefficient to the combustion zone (ft^2/sec)
DMPBDT	Mass rate of change of lead in lead layer (lb/sec)
DP1,..DP3	Increase in cell gas pressure due to each injection (psi)
DTBDT(i)	Concrete floor temperature rate of change, node i (deg. F./sec)
DTCDT(i)	Concrete wall temperature rate of change, node i (deg. F./sec)
DTMIN	User defined minimum time step length (sec)
DT1..DT5	$X/(dx/dt) * RELERR$, used in determining the time step length (sec)
DT1	X = Lithium pool temperature
DT2	X = Primary gas temperature
DT3	X = Primary wall liner temperature
DT4	X = Combustion rate
DT5	X = Combustion zone temperature*.05
DYNAMI	Subroutine used in controlling integration loops
D1	Kinematic viscosity of the cell gas (squared) at the film temperature (ft^4/sec^2), See related film temperature T1.
EFILM	Film depth of depleted zone above combustion zone (in)
EMCONC	Thermal emissivity of concrete
EMCZ	Thermal emissivity of combustion zone
EMF	Used in fixing minimum emissivity of the lithium pool (.9 in code)
EMGP	Thermal emissivity of primary gas (minimum of .005 in code)
EMGPF	Constant used in determining EMGP, usually chosen at .04
EMGS	Thermal emissivity of secondary gas (minimum of .005 in code)
EMINS	Thermal emissivity of pan insulation
EMLI	Thermal emissivity of lithium pool
ESCR	Heat removal rate by emergency space cooling (BTU/sec)
ESCTIN	Time after spill when ESCR begins (sec)
ESTAIR	Thermal emissivity of the cell steam-air mixture (without aerosols)
ESTLFP	Thermal emissivity of the primary floor liner
ESTLFS	Thermal emissivity of the secondary floor liner
ESTLWP	Thermal emissivity of the primary wall liner
ESTLWS	Thermal emissivity of the secondary wall liner
EW1	Thermal emissivity of water vapor at one atmosphere and cell temperature

EXHSTR	Rate of primary gas exhaust (lb/sec)
EXHSTV	Rate of primary gas exhaust (ft^3/sec)
EXX	Temporary variable used in calculating heat and mass diffusion coefficients (ft^{-3})
EX1	Used in calculating mass and heat transfer coefficients (ft^{-1}) See related film temperature T1.
FCO2P	Weight fraction of CO_2 in primary gas
FCO2S	Weight fraction of CO_2 in secondary gas
FCT1,FCT2 FCT3	Fraction of nitrogen in each injection (by number)
FF1,FF2	Used in heat balance equations for spray fire
FMLEAK	Fraction of mass of gas leaked out of primary
FMLEFT	Fraction of mass of gas remaining in containment
FNIP	Weight fraction of nitrogen in primary gas
FNIS	Weight fraction of nitrogen in secondary gas
FOUTP	Loss rate of primary gas which is either exhausted or changes cells
FOUTS	Loss rate of secondary gas which is either exhausted or changes cells
FOUTT	Total loss rate from outermost gas cell (FOUTS+LEAK)
FOXP	Weight fraction of oxygen in primary gas
FOXS	Weight fraction of oxygen in secondary gas
FPG	Radiative view factor from lithium pool to primary gas (1.0 w/o pan)
FPW	Radiative view factor from lithium pool to primary wall liner
FRA	Fraction of combustion products evolved into cell gas
FWAP	Weight fraction of water vapor in primary gas
FWAS	Weight fraction of water vapor in secondary gas
GAP	Air gap between floor liner and concrete (ft)
GAMMA	Ratio of specific heats c_p/c_v (set to 1.4 in the code)
GIN	Acceleration due to gravity (32.2 ft/sec^2)
GRPR	Product of the Grashof and Prandtl numbers of the water pool-gas boundary
GRPRF	Product of the Grashof and Prandtl numbers of the water pool
HGWP	Heat transfer coefficient between primary wall and gas (BTU/ft^2 sec deg. F.)
HA	Heat transfer coefficient between exterior wall and ambient (BTU/ft^2 sec deg. F.)
HAMF	Heat transfer coefficient between exterior floor and ambient (BTU/ft^2 sec deg. F.)
HB	Heat transfer coefficient between lithium pool and primary gas (BTU/ft^2 sec deg. F.)
HBINF	Equilibrium value of HB
HCOND	Total heat transfer coefficient between the water pool and the cell gas (BTU/ft^2 sec deg. F.)
HEHCP	Heat transfer coefficient between primary extraneous heat capacity and primary gas (BTU/ft^2 sec deg. F.)
HEHCS	Heat transfer coefficient between secondary extraneous heat capacity and secondary gas (BTU/ft^2 sec deg. F.)
HEVAP	Heat flux to water pool from gas mixture due to evaporation (BTU/sec-ft^2)

HF	Mass transport coefficient to the lithium pool (ft/sec)
HFG	Latent heat of vaporization of water vapor in the primary gas (BTU/lb)
HFG2	Latent heat of vaporization of water vapor in the secondary gas (BTU/lb)
HFINF	Equilibrium value of HF (BTU/ft^2 sec deg. F.)
HFPGP	Heat transfer coefficient between primary floor liner and primary gas (BTU/ft^2 sec deg. F.)
HFPGAS	Heat transfer coefficient between primary floor liner and secondary gas (BTU/ft^2 sec deg. F.)
HFSGAS	Heat transfer coefficient between secondary floor liner and secondary gas (BTU/ft^2 sec deg. F.)
HINxxx	Heat transfer coefficients are determined by LITFIRE as indicated in section II.2.2. The coefficients C are indicated in the code as HINxxx and are dimensionless
HINECP	Correlation for HEHCP
HINECS	Correlation for HEHCS
HINFAM	Correlation for HAMF
HINFGS	Correlation for HFPGAS
HINFSG	Correlation for HFSGAS
HINGPF	Correlation for HFPGP
HINGSP	Correlation for HGWP
HINGSS	Correlation for HSEC
HINJ	Specific enthalpy of steam injected to primary cell (BTU/lb)
HINJ2	Specific enthalpy of steam injected to secondary cell (BTU/lb)
HINPS	Correlation for HWPGAS
HINSAM	Correlation for HA
HLP	Specific enthalpy of the primary liquid water pool (BTU/lb)
HLP2	Specific enthalpy of the secondary liquid water pool (BTU/lb)
HLPFLR	Heat transfer coefficient between the liquid water pool and the secondary floor liner (BTU/ft^2 sec deg. F.)
HPAN	Heat transfer coefficient between the pan and primary gas (BTU/ft^2 sec deg. F.)
HPRIME	Increased value of HSENS due to a large condensation/evaporation rate (BTU/ft^2 sec deg. F.)
HRATIO	Molar fraction of hydrogen in the primary gas
HRAT2	Molar fraction of hydrogen in the secondary gas
HSAT	Specific enthalpy of saturated liquid water at cell pressure (BTU/lb)
HSEC	Heat transfer coefficient between secondary floor liner and secondary gas (BTU/ft^2 sec deg. F.)
HSENS	Sensible heat transfer coefficient from the water pool to the vapor mixture (BTU/ft^2 sec deg. F.)
HTCPGP	Heat capacity of the primary gas (BTU/deg. F.)
HTCPGS	Heat capacity of the secondary gas (BTU/deg. F.)
HU(x,y)	Chart of Uchida heat transfer coefficients vs. MR
HUCH	Uchida heat transfer coefficient for condensing steam (BTU/ft^2 sec deg. F.)

HUM	Initial relative humidity of the primary cell (1.0=100%)
HUM2	Initial relative humidity of the secondary cell (1.0=100%)
HWPGAS	Heat transfer coefficient between primary wall liner and secondary gas (BTU/ft^2 sec deg. F.)
HWV	Specific enthalpy of water vapor in the primary gas (BTU/lb)
HWV2	Specific enthalpy of water vapor in the secondary gas (BTU/lb)
I	General purpose DO loop counter
IAM	DO loop counter for wall and floor concrete node initialization
IB	DO loop counter for floor concrete iterations
INIT	Initializing subroutine for integrations
INJEC1..3	Flags for gas injection. INJEC*n* indicates the injection has occurred.
INTDSx	Interpolation factor used in reading steam tables
INTGRL	Arithmetic statement function for finding integrals
IPAGE	Number of lines of output between headings
IPASS	Used during integration routine to tell INTGRL to perform certain special functions during the first two executions of the section
KAIR	Thermal conductivity of non-condensible gas (air) (BTU/sec ft deg. F.)
KAIRB	Thermal conductivity of air at the water pool-gas boundary (BTU/sec ft deg. F.)
KAIRG	Thermal conductivity of air in a cell gas (BTU/sec ft deg. F.)
KB	Water condensation mass transfer coefficient (lb mol/sec-ft^2)
KBNDRY	Thermal conductivity of the water pool-gas boundary (BTU/sec ft deg. F.)
KLEAK	Leak rate constant from containment (in/lb$^{.5}$ sec) Note: units have been inferred from the code and may not be correct. Reference value: $2.588 \cdot 10^{-9}$
KCON	Thermal conductivity of concrete (input as BTU/hr ft deg. F.)
KFILM	Thermal conductivity of pool/combustion zone film (BTU/sec ft deg. F.)
KGAP	Thermal conductivity of the gap between the liner and concrete (BTU/hr ft deg. F.)
KH2O	Thermal conductivity of water vapor (BTU/sec ft deg. F.)
KH2OB	Thermal conductivity of water vapor at the water pool-gas boundary (BTU/sec ft deg. F.)
KH2OG	Thermal conductivity of water vapor in a cell gas (BTU/sec ft deg. F.)
KIN1	Thermal conductivity of inner insulation layer (BTU/hr ft deg. F.)
KIN2	Thermal conductivity of outer insulation layer (BTU/hr ft deg. F.)
KPAN	Thermal conductivity of the pan (BTU/hr ft deg. F.)
KSTLFP	Thermal conductivity of the primary floor liner (BTU/hr ft deg. F.)
KSTLFS	Thermal conductivity of the secondary floor liner (BTU/hr ft deg. F.)
KSTLWP	Thermal conductivity of the primary wall liner (BTU/hr ft deg. F.)
KSTLWS	Thermal conductivity of the secondary wall liner (BTU/hr ft deg. F.)
KWAT	Thermal conductivity of liquid water (BTU/sec ft deg. F.)
L	Thickness of a wall concrete node (ft)
L1	Thickness of a floor concrete node (ft)
LEAK	Gas leakage rate from outermost cell (sec^{-1})

LEAKO	Initial gas leakage rate from outermost cell ($\sec^{-1}$)
LIBP	Lithium consumed in pool fire (lb)
LIL	Amount of lithium remaining in pool—limited to LIT/10 for numerical stability during calculations
LILC2	Amount of Li_2C_2 in the pool (lb)
LILCA	Amount of Li_2CO_3 in the pool (lb)
LILCAR	Amount of carbon in the pool (lb)
LILNI	Amount of Li_3N in the pool (lb)
LILOX	Amount of Li_2O in the pool (lb)
LILP	True amount of lithium in the pool (lb)
LIS	Amount of lithium consumed in the spray fire (lb)
LIT	Initial mass of lithium in the pool (lb)
MAIP	Initial mass of inert gas in the primary gas (lb)
MAIS	Initial mass of inert gas in the secondary gas (lb)
MAIRP	Mass of primary non-condensible gas (lb)
MAIRS	Mass of secondary non-condensible gas (lb)
MAP	Mass of inert gas in the primary gas (lb)
MAS	Mass of inert gas in the secondary gas (lb)
MBOIL	Mass of water boiled off from the water pool in one time step (lb)
MCO2IP	Initial mass of carbon dioxide in the primary gas (lb)
MCO2IS	Initial mass of carbon dioxide in the secondary gas (lb)
MCO2P	Mass of carbon dioxide in the primary gas (lb)
MCO2S	Mass of carbon dioxide in the secondary gas (lb)
MCONDE	Condensation rate of water on an extraneous heat capacity (lb/sec)
MCONDF	Condensation rate of water on the secondary floor liner (lb/sec)
MCONDP	Condensation rate of water on the pan insulation (lb/sec)
MCONDW	Condensation rate of water on a wall liner (lb/sec)
MCONFP	Condensation rate of water on the primary floor liner from the secondary gas (lb/sec)
MCONWP	Condensation rate of water on the primary wall liner from the secondary gas (lb/sec)
MFAB	Molar fraction of air at the water pool-gas boundary
MFAG	Molar fraction of air in the cell gas
MFVB	Molar fraction of water vapor at the water pool-gas boundary
MFVG	Molar fraction of water vapor in the cell gas
MGB	Molecular weight of the mixture at the water pool-gas boundary
MH2P	Mass of hydrogen in the primary gas (lb)
MH2S	Mass of hydrogen in the secondary gas (lb)
MINJR	Mass flow rate of steam injected to primary cell (lb/sec)
MINJR2	Mass flow rate of steam injected to secondary cell (lb/sec)
MLC2P	Mass of lithium carbide in the primary gas (lb)
MLC2S	Mass of lithium carbide in the secondary gas (lb)
MLC3IP	Initial mass of lithium carbonate in the primary gas (lb)
MLC3IS	Initial mass of lithium carbonate in the secondary gas (lb)

MLC3P	Mass of lithium carbonate in the primary gas (lb)
MLC3S	Mass of lithium carbonate in the secondary gas (lb)
MLEAD	Mass of lead in the lead layer above the lithium-lead pool (lb)
MLIHP	Mass of lithium-hydroxide in the primary gas (lb)
MLIHS	Mass of lithium-hydroxide in the secondary gas (lb)
MLINIP	Initial mass of lithium-nitride in the primary gas (lb)
MLINIS	Initial mass of lithium-nitride in the secondary gas (lb)
MLINP	Mass of lithium-nitride in the primary gas (lb)
MLINS	Mass of lithium-nitride in the secondary gas (lb)
MLIOH	Mass of LiOH produced (moles)
MLIOIP	Initial mass of lithium-oxide in primary gas (lb)
MLIOIS	Initial mass of lithium-oxide in secondary gas (lb)
MLIOP	Mass of lithium-oxide in primary gas (lb)
MLIOS	Mass of lithium-oxide in secondary gas (lb)
MNIINJ	Rate of nitrogen injection during a one minute interval (lb/sec)
MNIIP	Initial mass of nitrogen in the primary gas (lb)
MNIIS	Initial mass of nitrogen in the secondary gas (lb)
MNIP	Mass of nitrogen in the primary gas (lb)
MNIS	Mass of nitrogen in the secondary gas (lb)
MNINJ1..3	Mass of nitrogen injected in gas injection (lb)
MOINJ1..3	Mass of oxygen injected in gas injection (lb)
MOXINJ	Rate of oxygen injection during a one minute interval (lb/sec)
MOXIP	Initial mass of oxygen in the primary gas (lb)
MOXIS	Initial mass of oxygen in the secondary gas (lb)
MOXP	Mass of oxygen in the primary gas (lb)
MOXS	Mass of oxygen in the secondary gas (lb)
MR	Mass ratio of air to water vapor in the cell gas
MUAIR	Viscosity of air in the cell gas (lb/sec-ft)
MUAIRB	Viscosity of air at the water pool-gas boundary (lb/sec-ft)
MUAIRG	Viscosity of air in the cell gas (lb/sec-ft)
MUB	Viscosity of the mixture at the water pool-gas boundary (lb/sec-ft)
MUDIFF	Viscosity of the mixture at the lithium pool surface (lb/sec-ft)
MUV	Viscosity of water vapor in the cell gas (lb/sec-ft)
MUVB	Viscosity of water vapor at the water pool-gas boundary (lb/sec-ft)
MUVCZ	Viscosity of water vapor at the lithium pool surface (lb/sec-ft)
MWL	Mass of liquid water in the primary water pool (lb)
MWL2	Mass of liquid water in the secondary water pool (lb)
MWLV	Mass of liquid water in the primary gas (lb)
MWLV2	Mass of liquid water in the secondary gas (lb)
MWLZ	Time rate of change of the mass of liquid water in the primary water pool (lb)
MWLZ2	Time rate of change of the mass of liquid water in the secondary water pool (lb)
MWV	Total mass of water in the primary gas (lb)

MWV2	Total mass of water in the secondary gas (lb)
MWVV	Mass of water vapor in the primary gas (lb)
MWVV2	Mass of water vapor in the secondary gas (lb)
MWVZ	Time rate of change of the total mass of water in the primary gas (lb)
MWVZ2	Time rate of change of the total mass of water in the secondary gas (lb)
N	Index used to transfer to section of subroutines, and as a DO loop counter
NAME(i)	Input variable containing program name and output headings
NL	Number of concrete wall nodes
NL1	Number of concrete floor nodes
NLM1	Number of concrete wall nodes minus one
NL1M1	Number of concrete floor nodes minus one
NS	Index used to control transfer to sections of the steam-air subroutines
NULV	Kinematic viscosity of water at the secondary floor liner (ft^2/sec)
OUTINT	Fraction of the outermost cell gas leaked to ambient
OVERPP	Primary cell overpressure (psig)
OVERPS	Secondary cell overpressure (psig)
OXLB	Mass of oxygen consumed in the fire (lb)
OXLBI	Mass of oxygen consumed in the spray fire (lb)
OXLFS	Mass of oxygen remaining after the spray fire (lb)
PAP	Primary gas pressure (psia)
PAPZER	Initial primary gas pressure (psia)
PAS	Secondary gas pressure (psia)
PASZER	Initial secondary gas pressure (psia)
PERCEN	Molecular percentage of lithium-peroxide (vs. monoxide) formed during combustion
PH2O	Partial pressure of water vapor in the primary gas (psia)
PH2O2	Partial pressure of water vapor in the secondary gas (psia)
PH2OB	Partial pressure of water vapor at the primary water pool-gas boundary (psia)
PH2OB2	Partial pressure of water vapor at the secondary water pool-gas boundary (psia)
PHASE	=1 if primary cell is saturated, =2 if superheated
PHASE2	=1 if secondary cell is saturated, =2 if superheated
PHIA	Baroczy two-phase flow correction factor at the lithium pool surface
PHIW	Baroczy two-phase flow correction factor at the lithium pool surface
PHIAWB	Baroczy two-phase flow correction factor at the water pool surface
PHIWAB	Baroczy two-phase flow correction factor at the water pool surface
PLIV	Partial pressure of lithium vapor (psia)
PRSC	Prandtl number divided by the Schmidt number
PSAT	Saturation pressure of water at the primary gas temperature (psia)
PSAT2	Saturation pressure of water at the secondary gas temperature (psia)
PYU	Used in setting the time step length calculated from the heat conduction rate to the pan or primary floor liner from the lithium pool
PZEROP	Primary gas pressure after the spray fire (psia)
QCxxx	Heat of combustion (BTU/lb Li)
QCC	Li-CO_2 producing lithium carbonate reaction

QCCON(	,oncrete reaction
QCN	Nitrogen reaction
QCO	Oxygen reaction
QCO1	Monoxide reaction
QCO2	Peroxide reaction
QCW	Water vapor reaction
QIN	Heat addition to primary gas from spray fire (BTU)
QL2C2	Heat of combustion for the carbon reaction (BTU/lb Li)
QLFLR	Heat flux from the secondary water pool to the secondary floor liner (BTU/ft^2)
QLIOH	Heat of fusion of LiOH (BTU/lb mol)
QOUT1..5	Used in heat balance equations for the spray fire (BTU)
QQQ	Used as a counter in the lithium transfer simulation
QRADxx	Indicates a radiative heat flow (BTU/sec)
QRADB	Floor liner (or pan) to ambient or floor concrete
QRADC	Wall liner to ambient or wall concrete
QRADCG	Pan to primary gas
QRADFG	Primary floor liner to secondary gas
QRADFS	Primary floor liner to secondary floor liner
QRADG	Combustion zone (or Li pool without combustion) to primary gas
QRADP	Combustion zone to lithium pool (only during combustion)
QRADPG	Primary wall liner to secondary gas
QRADPS	Primary wall liner to secondary wall liner
QRADS	Pan to primary floor liner
QRADW	Combustion zone (or lithium pool) to primary wall liner
QU	Heat flux from the primary gas to primary wall liner due to condensation (BTU/sec-ft^2)
QVAP	Heat of vaporization of lithium (BTU/lb)
QVEHC	Heat flux from the cell gas to the extraneous heat capacity due to condensation (BTU/sec-ft^2)
QVPAN	Heat flux from the primary gas to the pan insulation due to condensation (BTU/sec-ft^2)
QVSEC	Heat flux from the secondary gas to secondary wall liner due to condensation (BTU/sec-ft^2)
QVFLR	Heat flux from the cell gas to cell floor liner due to condensation (BTU/sec-ft^2)
QVFPGS	Heat flux from the secondary gas to primary floor liner due to condensation (BTU/sec-ft^2)
QVWPGS	Heat flux from the secondary gas to primary wall liner due to condensation (BTU/sec-ft^2)
RA	Mean radius of combustion product particles (microns)
RADxxx	'RAD' or 'R' indicates a temperature rate of change in one node due to radiative heat transfer to or from another (deg. F./sec)
'RAD'B	Floor liner to ambient or floor concrete

'RAD'C	Wall liner to ambient or wall concrete
'RAD'CB	Floor concrete from floor liner
'RAD'CC	Wall concrete from wall liner
RAIR	Gas constant for primary non-condensible gas (RIN/XMOLP)
RAIR2	Gas constant for secondary non-condensible gas (RIN/XMOLS)
RBREAK	Temperature rate of change of primary gas due to leakage (R/sec)
RC2	Li-CO_2 reaction rate inhibition factor (Li_2CO_3 reaction)
RC2LB	Rate of carbon consumption (lb/sec)
RCMBC2	Stoichiometric combustion ratio for lithium and carbon (lb Li/lb C)
RCMBCO	Stoichiometric combustion ratio for lithium and carbon dioxide for lithium carbonate producing reaction (lb Li/lb CO_2)
RCMBCS	Stoichiometric ratio of lithium consumed in Li_2CO_3 producing reaction to Li_2CO_3 produced (lb Li/lb Li_2CO_3)
RCMBH2	Stoichiometric combustion ratio for lithium and hydrogen (lb Li/lb H)
RCMBN	Stoichiometric combustion ratio for lithium and nitrogen (lb Li/lb N)
RCMBO	Stoichiometric combustion ratio for lithium and oxygen (lb Li/lb O)
RCMBO1	Stoichiometric combustion ratio for monoxide reaction (lb Li/lb O)
RCMBO2	Stoichiometric combustion ratio for peroxide reaction (lb Li/lb O)
RCMBW	Stoichiometric combustion ratio for lithium and water (lb Li/lb H_2O)
RCMCA1	Stoichiometric ratio of lithium consumed in Li-CO_2 reaction producing Li_2O and carbon to carbon produced (lb Li/lb C)
RCMCA2	Stoichiometric ratio of lithium consumed in Li-CO_2 reaction producing Li_2CO_3 and carbon to carbon produced (lb Li/lb C)
RCMCCO	Stoichiometric combustion ratio for lithium and carbon dioxide for lithium oxide producing reaction (lb Li/lb CO_2)
RCOLB	Rate of carbon dioxide consumption (lb CO_2/sec)
'R'CZG	Primary gas from combustion zone
'R'CZP	Lithium pool from combustion zone
'R'CZW	Primary wall liner from combustion zone
RELERR	Fraction of $X/(dx/dt)$ allowed as the maximum time step (sec)
'R'FPFS	Primary floor liner from secondary floor liner
'R'FPGAS	Primary floor liner from secondary gas
'R'FSFP	Secondary floor liner from primary floor liner
'R'GASFP	Secondary gas from primary floor liner
'R'GASPA	Pan to primary gas
'R'GLI	Lithium pool to primary gas (without combustion)
RHCON	Density of concrete (lb/ft^3)
RHINS	Density of insulation (lb/ft^3)
RHLEAD	Density of pure lead (lb/ft^3)
RHLI	Density of pure lithium (lb/ft^3)
RHLIX	Density of pure lithium (used in LC-2 lithium transfer simulation) (lb/ft^3)
RHOAIP	Initial density of primary non-condensible gas (lb/ft^3)
RHOAIS	Initial density of secondary non-condensible gas (lb/ft^3)

RHOAP	Density of primary non-condensible gas (lb/ft^3)
RHOAS	Density of secondary non-condensible gas (lb/ft^3)
RHOB	Density of the water pool-gas boundary layer (lb/ft^3)
RHOCAR	Density of carbon (lb/ft^3)
RHOLC2	Density of Li_2C_2 (lb/ft^3)
RHOLC3	Density of Li_2CO_3 (lb/ft^3)
RHOLIH	Density of LiOH (lb/ft^3)
RHOLIN	Density of Li_3N (lb/ft^3)
RHOLIO	Density of Li_2O (lb/ft^3)
RHOLIV	Density of lithium vapor above the pool (lb/ft^3)
RHOTOT	Total gas density at the lithium pool surface (lb/ft^3)
RHPAN	Density of lithium spill pan (lb/ft^3)
RHSFP	Density of primary floor liner (lb/ft^3)
RHSFS	Density of secondary floor liner (lb/ft^3)
RHSWP	Density of primary wall liner (lb/ft^3)
RHSWS	Density of secondary wall liner (lb/ft^3)
RIFxxx	Radiative interchange factor—used in radiative heat transfer
RIFCZG	Combustion zone and primary gas
RIFCZP	Combustion zone and primary wall liner
RIFFPS	Primary floor liner and secondary floor liner
RIFPAG	Pan to primary gas
RIFPAS	Pan to floor liner
RIFPG	Lithium pool to primary gas
RIFPGA	Primary wall liner to secondary gas
RIFPS	Primary wall liner to secondary wall liner
RIFPW	Lithium pool to primary wall liner
RIFSLC	Wall or floor liner to concrete
RIN	Universal gas constant (1545 ft-lbf/lb-mol-deg. F.)
RINP	Gas constant for the primary gas (RIN/XMOLP)
RINS	Gas constant for the secondary gas (RIN/XMOLS)
'R'LIG	Gas from lithium pool (without combustion)
'R'LIW	Wall liner from lithium pool (without combustion)
RN2	Lithium-nitrogen reaction rate inhibition factor
RNILB	Rate of nitrogen combustion (lb/sec)
RO2	Lithium-oxygen reaction rate inhibition factor
ROXLB	Rate of oxygen combustion (lb/sec)
'R'PAGAS	Primary gas from pan
'R'PANST	Primary wall liner from pan
RR	Function which generates the lithium-nitrogen reaction rate curve
'R'SPGS	Secondary gas from primary wall liner
'R'STPAN	Pan from primary wall liner
RWALB	Rate of water vapor consumption (lb/sec)
'R'WLI	Lithium pool from primary wall liner (without combustion)
'R'WPGAS	Primary wall liner from secondary gas

'R'WPWS	Primary wall liner from secondary wall liner
'R'WSWP	Secondary wall liner from primary wall liner
SAT(x,y)	Saturated steam table
SFLCR	Heat removal rate by emergency cooling of floor liner (BTU/sec)
SFLEND	Time after spill when SFLCR ends (sec)
SFLTIN	Time after spill when SFLCR begins (sec)
SH(x,y,z)	Superheated steam table
SIGMA	Stefan-Boltzmann constant ($1.713 \cdot 10^{-9}$ BTU/ft^2-hr-deg. R.^{4})
SPILL	Total mass of lithium spilled (lb)
SPRAY	Mass fraction of lithium consumed in the spray fire
STICK	Rate at which aerosols are removed from the primary cell due to sticking to the wall. If STICK $>$ 1.0, execution stops. STICK may be reduced by increasing BETA.
STMFAC	Density weighting factor for calculating steam-air mixture properties
STMIN	Time to begin steam injection to primary cell (sec)
STMIN2	Time to begin steam injection to secondary cell (sec)
STMOUT	Time to end steam injection to primary cell (sec)
STOUT2	Time to end steam injection to secondary cell (sec)
TA	Ambient temperature (deg. R.)
TAU	Time constant for transient natural convection
TAUCZ	Radiative transmissivity used to model pool-combustion zone coupling rather than (1.-EMCZ)
TAVE	Average of secondary floor and secondary water pool temperature (deg. R.)
TAVHI	Variable used to read steam tables
TAVLO	Variable used to read steam tables
TB(i)	Temperature of ith node of concrete floor (deg. R.)
TBIC(i)	Initial temperature of ith node of concrete floor (deg. R.)
TxxxxF	Corresponding temperature to Txxxx in Fahrenheit
TBLOW	Inert gas inlet temperature (deg. R.)
TC(i)	Temperature of ith node of concrete wall (deg. R.)
TCIC(i)	Initial temperature of ith node of concrete wall (deg. R.)
TCIGNI	Ignition temperature of lithium-concrete reaction (deg. R.)
TCON	Concrete combustion zone temperature (deg. R.)
TCZ	Combustion zone temperature (deg. R.)
TCZI	Initial combustion zone temperature (deg. R.)
TE	Equilibrium temperature resulting from spray fire (deg. R.)
TEHCP	Primary extraneous heat capacity temperature (deg. R.)
TEHCS	Secondary extraneous heat capacity temperature (deg. R.)
TEHCZP	Initial primary extraneous heat capacity temperature (deg. R.)
TEHCZS	Initial secondary extraneous heat capacity temperature (deg. R.)
TET1	Used in calculating thermal conductivity of inner pan insulation See KIN1
TET2	Used in calculating thermal conductivity of outer pan insulation See KIN2

TEZ	Average of combustion zone temperature and lithium pool temperature Used in test for combustion (deg. R.)
TFEFF	Normalized temperature of combustion zone-lithium pool temperature (deg. R.)
TFHI	Variable used to read steam tables
TFLO	Variable used to read steam tables
TFS	Secondary floor liner temperature (deg. R.)
TGP	Primary gas temperature (deg. R.)
TGPZER	Initial primary gas temperature (deg. R.)
TGS	Secondary gas temperature (deg. R.)
TGSZER	Initial secondary gas temperature (deg. R.)
THFC	Thickness of concrete floor (ft)
THFP	Thickness of primary floor liner (ft)
THFS	Thickness of secondary floor liner (ft)
THI	Temporary variable used to read steam tables
THKIN1	Thickness of inner pan insulation (ft)
THKIN2	Thickness of outer pan insulation (ft)
THKPAN	Thickness of spill pan (ft)
THPB	Thickness of lead layer above the lithium-lead pool (ft)
THWC	Thickness of concrete wall (ft)
THWP	Thickness of primary wall liner (ft)
THWS	Thickness of secondary wall liner (ft)
TIME	Time elapsed after spill has occurred (sec)
TIMEF	User defined time to stop execution of the code (sec)
TIMEO	Time at which code prints output data to a file (sec)
TINS1	Inner pan insulation layer temperature (deg. R.)
TINS1I	Initial inner pan insulation layer temperature (deg. R.)
TINS2	Outer pan insulation layer temperature (deg. R.)
TINS2I	Initial outer pan insulation layer temperature (deg. R.)
TLEAD	Temperature of the lead layer above the Li-Pb pool (deg. R.)
TLEADI	Initial temperature of the lead layer above the Li-Pb pool (deg. R.)
TLI	Lithium pool temperature (deg. R.)
TLIBS	Lithium pool temperature before spray fire (deg. R.)
TLII	Initial lithium pool temperature (deg. R.)
TLO	Temporary variable used to read steam tables
TLP	Temperature of the primary liquid pool (deg. R.)
TMELT	Melting temperature of lithium (deg. R.)
TO	Primary gas temperature before spray fire (deg. R.)
TONE,TTWO,TTHREE	Time at which each injection occurs (sec)
TPAN	Pan temperature (deg. R.)
TPANZO	Initial pan temperature (deg. R.)
TSAT	Saturation temperature of water based on its partial pressure (deg. F.)
TSFP	Primary floor liner temperature (deg. R.)
TSFPI	Initial primary floor liner temperature (deg. R.)

TSFSI	Initial secondary floor liner temperature (deg. R.)
TSP	Primary wall liner temperature (deg. R.)
TSPZER	Initial primary wall liner temperature (deg. R.)
TSS	Secondary wall liner temperature (deg. R.)
TSSZER	Initial secondary wall liner temperature (deg. R.)
TVAP	Vaporization temperature of lithium (deg. R.)
T1	Film temperature between primary gas and lithium pool (deg. R.)
T2,T3	Temporary variables used in setting up steam table
UA	Internal energy of non-condensible gas in a cell (BTU)
UGPB	Specific internal energy of saturated water vapor at boiling (BTU/lb)
UL	Internal energy of the primary water pool (BTU)
UL2	Internal energy of the secondary water pool (BTU)
ULP	Specific internal energy of the primary water pool (BTU/lb)
ULP2	Specific internal energy of the secondary water pool (BTU/lb)
ULPB	Specific internal energy of liquid needed for boiling to occur (BTU/lb)
ULZ	Time rate of change of UL (BTU/sec)
ULZ2	Time rate of change of UL2 (BTU/sec)
USUBA	Heat transfer coefficient between the outermost containment node and the ambient (BTU/sec-ft^2-deg. F.)
UV	Internal energy of the primary gas (BTU)
UV2	Internal energy of the secondary gas (BTU)
UVZ	Time rate of change of the internal energy of the primary gas (BTU/sec)
UVZ2	Time rate of change of the internal energy of the secondary gas (BTU/sec)
UWV	Specific internal energy of water vapor in the primary gas (BTU/lb)
UWV2	Specific internal energy of water vapor in the secondary gas (BTU/lb)
VA	Specific volume of non-condensible primary gas (ft^3/lb)
VAB	Specific volume of non-condensible gas at the water pool-gas boundary (ft^3/lb)
VCONC	Volume of concrete in the first node of concrete (ft^3)
VHI	Variable used to read steam tables
VG	Specific volume of water in the primary gas (ft^3/lb)
VG2	Specific volume of water in the secondary gas (ft^3/lb)
VL	Volume of the primary water pool (ft^3)
VL2	Volume of the secondary water pool (ft^3)
VLO	Variable used to read steam tables
VLP	Specific volume of the primary water pool (ft^3/lb)
VLP2	Specific volume of the secondary water pool (ft^3/lb)
VLPF	Specific volume of saturated liquid water at the secondary floor liner temperature (ft^3/lb)
VLPV	Specific volume of saturated liquid water for heat transfer between the secondary floor liner and the secondary water pool (ft^3/lb)
VP	Volume of primary cell (ft^3)
VS	Volume of secondary cell (ft^3)
VSB	Specific volume of water vapor at the water pool-gas boundary (ft^3/lb)
VST	Specific volume of water vapor at the lithium pool surface (ft^3/lb)

VVB	Specific volume of saturated water vapor at the primary water pool temperature (ft^3/lb)
VVB2	Specific volume of saturated water vapor at the secondary water pool temperature (ft^3/lb)
VVG	Specific volume of saturated water vapor in the primary gas (ft^3/lb)
VVG2	Specific volume of saturated water vapor in the secondary gas (ft^3/lb)
VVT	Temporary variable used to determine steam properties for condensation
WAB	Mass fraction of inert gas in the flooding gas
WAP	Mass fraction of inert gas in the primary gas
WAS	Mass fraction of inert gas in the secondary gas
WATER	Amount of water that should be left in the top concrete node according to the correlation being used (lb/ft^3)
WCB	Mass fraction of carbon dioxide in the flooding gas
WCP	Mass fraction of carbon dioxide in the primary gas
WCO2S	Mass fraction of carbon dioxide in the secondary gas
WN2B	Mass fraction of nitrogen in the flooding gas
WN2P	Mass fraction of nitrogen in the primary gas
WN2S	Mass fraction of nitrogen in the secondary gas
WO2B	Mass fraction of oxygen in the flooding gas
WO2P	Mass fraction of oxygen in the primary gas
WO2S	Mass fraction of oxygen in the secondary gas
WWAB	Mass fraction of water vapor in the flooding gas
WWAP	Mass fraction of water vapor in the primary gas
WWAS	Mass fraction of water vapor in the secondary gas
XALLOY	Atom percent of lithium in lithium-lead pool
XAM	Logarithmic mean molar fraction of air (see MFAB and MFAG)
XBLOW	Used in conjunction with IBLOW
XESC	Used in conjunction with IESC
XINJ	Indicates whether steam injection to the primary is in effect
XINJ2	Indicates whether steam injection to the secondary is in effect
XLI	Mass fraction of lithium in lithium-lead pool
XLIDOT	Mass flow rate of lithium through the lead layer above the Li-Pb pool (lb/sec)
XMAIRP	Amount of non-condensible primary gas after spray fire (lb-mol)
XMAIRS	Amount of non-condensible secondary gas after spray fire (lb-mol)
XMDOT	Mass flow rate of gas between primary and secondary cells (lb/sec)
XMEHCP	Mass of primary extraneous heat capacity (lb)
XMEHCS	Mass of secondary extraneous heat capacity (lb)
XMH20I	Initial mass of water in concrete (lb)
XMOLP	Molecular weight of non-condensible primary gas (lb/lb-mol)
XMOLS	Molecular weight of non-condensible secondary gas (lb/lb-mol)
XMOLA	Molecular weight of containment inert gas (lb/lb-mol)
XMOLAB	Molecular weight of flooding inert gas (lb/lb-mol)
XSFL	Indicates whether emergency floor cooling is currently in effect

YALIG	Effective thermal admittance between the pool and primary gas (BTU/sec-deg. F.)
YAPCZ	Effective thermal admittance between the pool and combustion zone (BTU/sec-deg. F.)
YPAGAS	Effective thermal admittance between the pan and primary gas (BTU/sec-deg. F.)
ZLI	Thickness of the lithium pool (ft)
ZP	Used to determine EMLI if EMLI < 0.9
ZZxxxx	Temperature rate of change of a node (deg. R./sec)
ZZ1	Lithium pool
ZZ2	Lithium spill pan
ZZ3	Secondary cell gas
ZZ4	Primary cell gas
ZZ5	Primary wall liner
ZZ6	Combustion zone
ZZ7	Primary floor liner
ZZ8	Inner insulation layer
ZZ9	Outer insulation layer
ZZ99	Change in combustion rate with respect to time (lb Li/sec^2ft^2)
'ZZ'EP	Primary extraneous heat capacity
'ZZ'ES	Secondary extraneous heat capacity
'ZZ'FS	Secondary floor liner
'ZZ'PB	Lead layer above Li-Pb pool
'ZZ'S	Secondary wall liner

PROGRAM DECISION FLAGS

IAROSL	=1 to use aerosol removal from containment by sticking option
IBLOW	=1 Containment flooding with inert gas
	=0 No containment flooding
ICMB	=0 No oxygen left after spray fire
	=1 Still oxygen left after spray fire (initially =1, reset by code)
ICNI	=0 Nitrogen reactions not possible
	=1 Nitrogen reactions possible
ICO2I	=1 to use pure CO_2 atmosphere
ICZ	=0 Combustion zone model not used
	=1 Combustion zone model used
IESC	=1 to use emergency space cooling option
IFLAG2	=1 to use two-cell geometry option
IFLAGB	=1 to use lithium-lead option
IFLAGC	=1 to use concrete combustion option
IFLAGCO	=1 to use pure CO_2 atmosphere option
IFLAGD	=1 to use layered lithium-lead pool option
IFLAGF	=1 to use floor concrete option
IFLAGISI	=1 to enter input data in SI units

IFLAGP	=1 to use pan option
IFLAGS	=1 to use dry gas injection option
IFLAGT	=1 to use steam-air mixture option
IFLAGU	=1 to get output in SI units
IFLAGW	=1 to use wall concrete option
ILIT	=0 No lithium left to burn
	=1 Lithium left to burn
IMETH	=1 Runge-Kutta method of integration used
	=3 Simpson's Rule method of integration used
ISFLC	=1 to use emergency floor cooling option
ISWICH	=0 Crack size remains constant
	=1 Crack size reset to zero after primary and secondary cell gas pressure equilibrate (note: this should not be used when either cell is small compared to the other or the volume of gas being consumed by the fire)

OPTION AND LOGICAL DECISION FLAGS

FLAG2 = .TRUE.	Two cell geometry
FLAGAS = .TRUE.	Injection of dry gas during run
FLAGC = .TRUE.	Concrete combustion
FLAGCO = .TRUE.	Pure CO_2 containment atmosphere
FLAGD = .TRUE.	Concrete combustion has stopped (set by code)
FLAGDF = .TRUE.	Lithium-lead layered pool combustion model
FLAGF = .TRUE.	Floor concrete
FLAGISI = .TRUE.	Code accepts input in SI units
FLAGL = .TRUE.	LILP is fixed at a minimum (set by code)
FLAGM = .TRUE.	Sonic gas flow between cells (set by code)
FLAGN = .TRUE.	Indicates first run through a subroutine (set by code)
FLAGPB = .TRUE.	Lithium-lead combustion
FLAGPN = .TRUE.	Pan option
FLAGSI = .TRUE.	Code prints output in SI units
FLAGST = .TRUE.	Steam-air mixture in containment
FLAGW = .TRUE.	Wall concrete

Appendix C

Troubleshooting (Dube 1978)

"There exists no large computer code which runs perfectly 100% of the time."

-ANONYMOUS-

The user of this code may encounter problems while trying to execute LITFIRE. The most common error statement is generated by the computer itself: *DIVIDE BY ZERO* and stops the code. This indicates that an attempt was made to divide by zero or something very close to zero. The first step when encountering an error message is to check the output file *out1.dat* and ensure that the input was properly entered into the code. If it was, the error may occur if the value of EXHSTV is too high or LILP is too low. This problem may be mitigated to an extent by reducing DTMIN, the minimum time step length, although this will increase computation time.

Another common error message is generated by LITFIRE: *EXX IS NEGATIVE — CANNOT TAKE ROOT* When this occurs, it indicates that the code is trying to take the square root of a negative number — the code has diverged numerically and the combustion zone temperature is negative. This may occur when the combustion rate CMBRH is very small (i.e.,$\ll$ 1.0 lb Li/hr-ft^2), the oxygen and nitrogen concentrations are low, or when the gas pressure is low. This problem occurs when ZZ6 and DELT are large enough to produce a negative TCZ. This problem may be solved by reducing the value of DELOUT to limit the time step size, so extrapolations of TCZ over a long time step do not cause problems. Unfortunately, this can increase computation time considerably. The variation of the combustion zone temperature over time is a good indicator of whether or not the code is running properly. During combustion, TCZ should be 100^o F. or more higher than TLI. Once combustion stops, TCZ should drop rapidly to a value just barely above TLI. (TLI is hypothetical at this point if the lithium has been consumed, but it is continuously calculated for numerical reasons.) If TCZ oscillates rapidly or falls below TLI, it is an indication of trouble. As stated earlier, this may be mitigated by decreasing the value of DELOUT.

If the statement: *LITHIUM TEMP. ABOVE BOILING POINT* occurs, this may indicate that the rate of change of the lithium pool temperature was very large. This may be due to the fact that TCZ has diverged to a very large value. This generally occurs as LILP nears zero, as the pool then has a lower heat capacity, and a sudden influx of energy would cause the pool to heat up rapidly.

Other messages indicate that the user is attempting to use incompatible options together, or that the code has been stopped because there is no point in continuing further (i.e. the lithium temperature has dropped below the melting point, or containment gas and pressure have returned to normal).

Other signs of trouble include a rapid drop in containment gas mass (MNIP, MOXP), or an oscillating combustion zone temperature, TCZ; combustion rate, CMBRH; or time step length DELT. In general, DELT should increase after the start of the run and then level off until combustion stops, when it may change more rapidly. Sudden large changes in DELT indicate that the temperature rates of change are varying rapidly, which usually should not be the case. If reducing DELT does not help solve the problem, print out values of the code quantities like ZZ5 and ZZ6 or UVZ and MWVZ to help find the problem.

Appendix D

Sample Input Data Files

INPUT DATA FILE UWMAK.W

```
 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
  20  20
  .10  .10  .10  .10  .10  .10  .10  .10  .10  .10
  .10  .10  .10  .10  .10  .10  .10  .10  .10  .10
  .10  .10  .10  .10  .10  .10  .10  .10  .10  .10
  .10  .10  .10  .10  .10  .10  .10  .10  .10  .10
   265000.00      50.00         0.1247      39.90
    538.0      25300.00      000.00         0.1199      0.00
      0.8333      2.0800        0.0020      0.015       0.00
      0.85        0.1189      30.00       497.500    12903.00       0.0197
      0.85        0.1189      30.00       497.500     5300.00       0.0197
      0.2         0.9960      33.80        30.00
      0.9         0.2250        0.0227    144.00
    124.00       86.9400     160.00         0.04        0.90        0.100
  18510.0         0.0       4080.0      13784.0
      0.8764      0.0          1.487        0.383       6.93
    816.60      2916.0      8431.0          0.0
      0.87        0.13         0.01
      0.12        0.12         0.112        0.07        0.07        0.07
      0.07        0.07         0.07
   4842.000    48280.00        0.000        0.050       5.0
   1400.0       1572.000    1572.00      1572.0       538.99     1392.0
     00.0001      0.2316       0.0000       0.0000      0.0000
00010000000.01000005000.00000000000.00600000005.00000000020.0000
```

INPUT DATA FILE UWMAK.X

```
8855700.00000000150.00000000014.70000000538.00000000538.00000000538.0000
     15.5000      0.00         0.232        0.0         522.0000      0.0
   1482.0   1696200.0      9709.00          0.6921       0.09
      0.85        0.1199      30.00       497.50    188164.00       0.0208
      0.85        0.1199      30.00       497.50     59038.00       0.0208
    350.0
```

INPUT DATA FILE UWMAK.Y

```
0000013.00000000490.00000000000.12000000010.00000000000.20000000000.9000
    535.50       35.29       16.50        14.15         0.000
      0.0157      0.1667       0.0833
      0.0350      9.3000     708.0000       0.1700    3315.0000    0000.0415
 6.45600E-08
```

```
INPUT DATA FILE UWMAK.Z

0000100.00000000000.00000000000.00000000004.00000000000.12470000535.0000
     24.0         0.0         310.0         325.0          0.00
```

```
INPUT DATA FILE STEAMOP.

0000600.000000004000.000000000000.000000001175.000
     20.00       800.00         50.0         1400.000
```

```
 INPUT DATA FILE HEAD.DAT

    THIS IS THE INPUT DATA FOR THE EXECUTION
     OF THE CODE LITFIRE
    THESE ARE THE OUTPUT VALUES CORRESPONDING
     TO THE PRIMARY CELL ENVIRONMENT
          TIME   DELT    TCZF      TLIF    TGPF     PAP      TSPF     TSFPF
    THESE ARE THE OUTPUT VALUES CORRESPONDING
     TO THE SECONDARY CELL ENVIRONMENT
    TIME    TGSF    TFSF     PAS       XMDOT     MOXS      MNIS       MCO2S
    THESE ARE THE OUTPUT VALUES CORRESPONDING
     TO THE PAN OUTPUT OPTION
           TIME       TLIF        TPANF        TINS1F       TINS2F        PAP
    THESE ARE ADDITIONAL OUTPUT VALUES CORRESPONDING
     TO THE PRIMARY CELL ENVIRONMENT
           TIME      MNIP      MOXP       MCO2P          RN2       RO2       LIBP
    THESE ARE THE OUTPUT VALUES CORRESONDING
     TO THE LITHIUM/LEAD DIFFUSION OPTION
           TIME     XLIDOT     TLEADF          MLEAD          THPB            ZLI
```

Appendix E

Sample Output Data Files

```
OUTPUT DATA FILE OUT2.DAT

     THESE ARE THE OUTPUT VALUES CORRESPONDING
     TO THE PRIMARY CELL ENVIRONMENT


          TIME   DELT     TCZF       TLIF    TGPF      PAP     TSPF     TSFPF
           0.0   0.01     504.78     500.33  600.33    0.00   600.33   600.33
         100.2   0.40     503.49     503.45  276.60    7.12   493.67   486.51
         200.1   0.80     495.00     494.93  295.82   14.75   425.79   477.89
         300.9   1.00     486.31     486.23  299.34   22.39   377.43   469.98
         400.6   1.00     481.56     477.68  300.03   29.87   344.27   462.18
         500.6   1.00     478.73     469.90  302.73   37.30   326.15   454.83
         600.6   1.00     478.00     463.54  306.52   44.81   312.44   448.72
         700.6   1.00     479.39     458.70  311.25   52.40   302.09   443.96
         800.6   1.00     482.90     455.44  316.09   59.98   294.50   440.64
         900.6   1.00     488.20     453.78  321.07   67.21   289.28   438.80
```

```
OUTPUT DATA FILE OUT3.DAT

 flow between primary and secondary has become sonic
    THESE ARE THE OUTPUT VALUES CORRESPONDING
    TO THE SECONDARY CELL ENVIRONMENT


   TIME     TGSF     TFSF     PAS       XMDOT        MOXS         MNIS          MCO2S
    0.0     25.89    25.89   101.38 0.3349E+01 0.1509E+06 0.4994E+06 0.0000E+00
  100.2     30.99    31.96   102.99 0.3374E+01 0.1507E+06 0.4989E+06 0.0000E+00
  200.1     35.43    37.04   104.38 0.3394E+01 0.1505E+06 0.4983E+06 0.0000E+00
  300.9     39.51    41.91   105.63 0.3413E+01 0.1504E+06 0.4977E+06 0.0000E+00
  400.6     43.22    46.48   106.76 0.3429E+01 0.1502E+06 0.4971E+06 0.0000E+00
  500.6     46.71    50.83   107.82 0.3444E+01 0.1500E+06 0.4966E+06 0.0000E+00
  600.6     50.01    54.96   108.80 0.3457E+01 0.1498E+06 0.4960E+06 0.0000E+00
  700.6     53.13    58.93   109.72 0.3470E+01 0.1497E+06 0.4954E+06 0.0000E+00
 flow between primary and secondary has returned to subsonic
  800.6     56.11    62.77   110.59 0.3435E+01 0.1495E+06 0.4948E+06 0.0000E+00
  900.6     58.95    66.50   111.42 0.3209E+01 0.1493E+06 0.4942E+06 0.0000E+00
```

```
OUTPUT DATA FILE OUT5.DAT

     THESE ARE ADDITIONAL OUTPUT VALUES CORRESPONDING
     TO THE PRIMARY CELL ENVIRONMENT


      TIME      MNIP        MOXP        MCO2P        RN2       RO2       LIBP
       0.0  0.1584E-01  0.4775E-02  0.0000E+00  0.00000   0.97117  0.0000E+00
     100.2  0.2588E+03  0.7819E+02  0.0000E+00  0.00000   0.97120  0.0000E+00
     200.1  0.5184E+03  0.1566E+03  0.0000E+00  0.00000   0.97120  0.0000E+00
     300.9  0.7818E+03  0.2362E+03  0.0000E+00  0.00000   0.97120  0.0000E+00
     400.6  0.1043E+04  0.3137E+03  0.0000E+00  0.23165   0.97115  0.3210E+01
     500.6  0.1300E+04  0.3855E+03  0.0000E+00  0.23322   0.97092  0.1973E+02
     600.6  0.1556E+04  0.4547E+03  0.0000E+00  0.23480   0.97070  0.4254E+02
     700.6  0.1810E+04  0.5210E+03  0.0000E+00  0.23642   0.97047  0.7237E+02
     800.6  0.2062E+04  0.5840E+03  0.0000E+00  0.23811   0.97023  0.1099E+03
     900.6  0.2299E+04  0.6400E+03  0.0000E+00  0.23994   0.96998  0.1559E+03
```

```
OUTPUT DATA FILE OUT9.DAT


 These outputs are the weights of reaction products in LB

    Time    Lilox       Lilni       Lilca       Lilc2       Lilcar     Mllop      Mlc3p

     0.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
   100.2 0.806E+00 0.842E+00 0.000E+00 0.000E+00 0.000E+00 0.424E-01 0.000E+00
   200.1 0.584E+01 0.610E+01 0.000E+00 0.000E+00 0.000E+00 0.307E+00 0.000E+00
   300.9 0.179E+02 0.187E+02 0.000E+00 0.000E+00 0.000E+00 0.942E+00 0.000E+00
   400.6 0.388E+02 0.406E+02 0.000E+00 0.000E+00 0.000E+00 0.204E+01 0.000E+00
   500.6 0.702E+02 0.739E+02 0.000E+00 0.000E+00 0.000E+00 0.369E+01 0.000E+00
   600.6 0.113E+03 0.120E+03 0.000E+00 0.000E+00 0.000E+00 0.595E+01 0.000E+00
   700.6 0.168E+03 0.182E+03 0.000E+00 0.000E+00 0.000E+00 0.885E+01 0.000E+00
   800.6 0.237E+03 0.260E+03 0.000E+00 0.000E+00 0.000E+00 0.125E+02 0.000E+00
   900.6 0.320E+03 0.356E+03 0.000E+00 0.000E+00 0.000E+00 0.168E+02 0.000E+00
```

```
OUTPUT DATA FILE OUT10.DAT


 These outputs are the reaction rates in gram Li/min-cm2

      Time     Cmbrhh      Cmbrnh      Cmbroh      Cmbrwh      Cmrcoh      Cmrc2h

       0.0 0.0000E+00 0.8092E-15 0.6031E-15 0.0000E+00 0.0000E+00 0.0000E+00
     100.2 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
     200.1 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
     300.9 0.0000E+00 0.6441E-03 0.4816E-03 0.0000E+00 0.0000E+00 0.0000E+00
     400.6 0.1815E-02 0.1041E-02 0.7742E-03 0.0000E+00 0.0000E+00 0.0000E+00
     500.6 0.2609E-02 0.1510E-02 0.1099E-02 0.0000E+00 0.0000E+00 0.0000E+00
     600.6 0.3498E-02 0.2043E-02 0.1455E-02 0.0000E+00 0.0000E+00 0.0000E+00
     700.6 0.4482E-02 0.2642E-02 0.1840E-02 0.0000E+00 0.0000E+00 0.0000E+00
     800.6 0.5565E-02 0.3312E-02 0.2253E-02 0.0000E+00 0.0000E+00 0.0000E+00
     900.6 0.6701E-02 0.4028E-02 0.2672E-02 0.0000E+00 0.0000E+00 0.0000E+00
```

Appendix F

Listing of the LITFIRE code

```
c -*- fortran -*-
c
c*************************************************************
c************** litfire.for:  a simulation of a lithium ******
c************** spill in a tokamak fusion reactor        *****
c*************************************************************
c
c                 lithium--steam reaction modeling included
c
c                 libp combustion modeling included
c
c                 akexx subroutine included
c
c   modeled with: taucz,emgp=1.0etc.,emgf is included,knit/klit.
c                    beta and stick
c                    seperate emissiviites and steel properties.
c                    new floor node in secodary.
c
c
      implicit real (i,k,l,m)
      logical flagw,flagf,flagl,flagpn,flagas,flagm,flag2,flagsi,flagn,
     .         flagc,flagpb,flagisi,flagdf,flagco,flagst
      integer iflagw,iflagf,iflagp,iflag2,iflags,iflagc,
     .              iflagu,iflagb,iblow,iesc,isflc,iswich,iarosl,
     .              iflagd,iflagisi,iflagco,iflagt,num,imeth,ipage,
     .              icz,icount,istore,inoin,ipass,
     .              i10,i11,i12,i13,i14,i15,i16,i17,i18,i19,i
      real intgrl,nulv
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .               rhli,spill,tli,tlii,zli
      common /lead/ cplead,klead,rhlead,mlipb,xalloy,atml,atmpb,cmbr
      common /pbpool/ dmpbdt,zzpb,mlead,tleadi,xwli,dflipb,xlidot,
     .                 thpb,tleadf,foo
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .                kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .               ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .               rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .               tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .               tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .               rair
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .                xic(101),zzz(501)
      common /injop/ dp1,dp2,dp3,mniinj,moxinj,time,vp
      common /panop/ ains,apan,bredth,clist,cpins,cppan,emgp,fpg,fpw,
     .                kpan,rhins,rhpan,thkin1,thkin2,thkpan,
     .                tins1,tins1f,tins1i,tins2,tins2f,tins2i,
     .                tpan,tpanf,tpanzo,zz2,zz4,zz8,zz9
      common /conop/ c8,cpcon,dtbdt(20),dtcdt(20),gap,kcon,kgap,
     .                l(20),l1(20),nl,nl1,qradb,radb,rhcon,
     .                sflcr,tb(20),tbf(20),tbic(20),tcf(20),
     .                tcic(20),thfc,thwc,tsfpl,tspzer,xsfl,qlflr,qvflr
      common /ccop/ cmbro,cracon,dcocz,h2left,qcconc,rcmbo,rcmbw,
     .               relese,tcigni,tcon,tconf,xmh2oi,zzc,zzd,zzdin,
     .               rcmbco,rcmbc2,rcmbcs,rcmca2,rcmca1,rcmcco
      common /secop/ aehcs,c11,c20,chs,cpehcs,cph2,cplih,cpwa,crack,
     .                foutp,fouts,foutt,hinfgs,hinfsg,hingss,hinps,kleak,
     .                leak,mairp,mairs,mais,mas,mh2s,mlihs,mlinis,mlins,
     .               mliois,mlios,mniis,mnis,moxis,moxs,mwais,
     .              mlc3s,mlc2s,mcs,mco2s,rholc3,rholc2,rhocar,
     .              mwas,pap,pas,paszer,ra,rbreak,rholih,
     .              rholin,rholio,rwpgas,tehcs,tehcsf,tehczs,tgsf,
     .          tfsf,tgszer,tssf,vs,xmdot,xmehcs,xmola,zz3,zzfs
      common /units/ aehcp,beta,chp,cmbrh,cpap,cpehcp,map,mnip,mco2p,
     .              moxp,mwap,papzer,qcn,qco,qco1,qco2,qcw,qvap,qcc,
     .              ql2c2,tcz,tczf,tczi,tehcp,tehcpf,tehczp,tgpf,
     .              tlif,tmelt,tsfpf,tspf,tvap,xmehcp
      common /pbdif/ cczp,cgli,clig,cpcz,cpmcz,dfilm,kfilm,pyup,
     .              qradp,rczp,rgli,rifczp,rifpg,rifpw,rlig,rwli,
     .              tlead,yapcz,zz6,dflvar
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .              ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .              phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +              mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .              xmolp,cell
      common /steam2/cpa2,cpvb2,emgs,hfg2,hlp2,htcpgs,hum2,mwl2,
     .              mwlv2,mwlz2,mwlzr2,mwv2,mwvv2,mwvz2,mwvzr2,
     .              phase2,ph2o2,ph2ob2,rair2,tlp2,ul2,ulp2,ulz2,
     .              ulzer2,uv2,uvz2,uvzer2,uwv2,vg2,vl2,vlp2,vvb2,
     .              vvg2,xg2,xmols,stmin2,stout2,xinj2,rhoas
      common /stmop/ minjr,minjr2,hinj,hinj2
      common /stmpn/ tsat,hsat,huch
      common /heat/  hingsp,hinecp
c
c      open(unit=1,file='head.dat',status='old')
c      open(unit=2,file='uwmak.w',status='old')
c      open(unit=3,file='uwmak.x',status='old')
c      open(unit=4,file='uwmak.y',status='old')
c      open(unit=5,file='uwmak.z',status='old')
c      open(unit=6,file='steamop.',status='old')
       call link("unit1=head.dat,read1//")
       call link("unit2=uwmak.w,read2//")
       call link("unit3=uwmak.x,read3//")
       call link("unit4=uwmak.y,read4//")
       call link("unit5=uwmak.z,read5//")
       call link("unit6=steamop,read6//")
c      open(unit=10,file='out1.dat',status='new')
c      open(unit=11,file='out2.dat',status='new')
c      open(unit=12,file='out3.dat',status='new')
c      open(unit=13,file='out4.dat',status='new')
c      open(unit=14,file='out5.dat',status='new')
c      open(unit=15,file='out6.dat',status='new')
c      open(unit=16,file='out7.dat',status='new')
c      open(unit=17,file='out8.dat',status='new')
c      open(unit=18,file='out9.dat',status='new')
c      open(unit=20,file='out10.dat',status='new')
c      open(unit=21,file='for021.dat',status='new')
c      open(unit=22,file='for022.dat',status='new')
c      open(unit=23,file='for023.dat',status='new')
c      open(unit=24,file='for024.dat',status='new')
c      open(unit=40,file='for040.dat',status='new')
c      open(unit=41,file='for041.dat',status='new')
c      open(unit=42,file='for042.dat',status='new')
c      open(unit=43,file='for043.dat',status='new')
c      open(unit=44,file='for044.dat',status='new')
c      open(unit=99,file='for099.dat',status='new')
       call link("unit10=(out1,text,create),print10//")
       call link("unit11=(out2,text,create),print11//")
       call link("unit12=(out3,text,create),print12//")
       call link("unit13=(out4,text,create),print13//")
       call link("unit14=(out5,text,create),print14//")
       call link("unit15=(out6,text,create),print15//")
      call link("unit16=(out7,text,create),print16//")
      call link("unit17=(out8,text,create),print17//")
      call link("unit18=(out9,text,create),print18//")
      call link("unit20=(out10,text,create),print20//")
      call link("unit21=(for021,text,create),print21//")
      call link("unit22=(for022,text,create),print22//")
      call link("unit23=(for023,text,create),print23//")
      call link("unit24=(for024,text,create),print24//")
      call link("unit40=(for040,text,create),print40//")
      call link("unit44=(for044,text,create),print44//")
      call link("unit99=(for099,text,create),print99//")
c
      call dropfile(0)
c***********************************************************************o
c                      input section                                    o
c***********************************************************************o
c   see litfire users guide for definitions and dimensions of input variables  o
c
c**********      read in title and headings      **********
c
      read (1,700)  (name(i),i=1,340)
  700 format(20a4)
c
c**********      read in flags and options      **********
c
c the next bunch of statements are here because of compile trouble at
c  livermore. hopefully this will be corrected soon. (1/25/82).
c
      read (2,701) iflagw,iflagf,iflagp,iflag2,iflags,iflagc,
     .             iflagu,iflagb,iblow,iesc,isflc,iswich,iarosl,
     .             iflagd,iflagisi,iflagco,iflagt
  701 format(1x,17(i1,1x))
      flagw=.false.
      flagf=.false.
      flagpn=.false.
      flag2=.false.
      flagas=.false.
      flagc=.false.
      flagsi=.false.
      flagpb=.false.
      flagdf=.false.
      flagisi=.false.
      flagco=.false.
      flagst=.false.
      if (iflagw .eq. 1) flagw=.true.
      if (iflagf .eq. 1) flagf=.true.
      if (iflagp .eq. 1) flagpn=.true.
      if (iflag2 .eq. 1) flag2=.true.
      if (iflags .eq. 1) flagas=.true.
      if (iflagc .eq. 1) flagc=.true.
      if (iflagu .eq. 1) flagsi=.true.
      if (iflagb .eq. 1) flagpb=.true.
      if (iflagd .eq. 1) flagdf=.true.
      if (iflagisi .eq. 1) flagisi=.true.
      if (iflagco .eq. 1) flagco=.true.
      if (iflagt .eq. 1) flagst=.true.
c
c**********      read in primary containment specifications      **********
c
      read (2,703) nl,nl1
      read (2,704) (l(i),i=1,nl)
      read (2,704) (l1(i),i=1,nl1)
      read (2,707) vp,chp,cpap,xmola
      read (2,702) tehczp,xmehcp,aehcp,cpehcp,hinecp
  702 format (6f12.4)
  703 format (i4,i4)
  704 format (10f5.3/10f5.3)
  706 format (7f12.4)
  707 format (f12.2,3f12.4)
c
c**********     read in parameters associated with     *********
c          outermost containment shell and concrete
c
      read (2,702) thwc,thfc,gap,kgap,kleak
      if (thwc .lt. 0.001) flagw=.false.
      if (thfc .lt. 0.001) flagf=.false.
c
c**********     read in physical constants     **********
c                  and emissivities
c
      read (2,702) estlwp,cpswp,kstlwp,rhswp,awp,thwp
      read (2,702) estlfp,cpsfp,kstlfp,rhsfp,afp,thfp
      read (2,702) emli,cpli,akli,rhli
      read (2,702) emconc,cpcon,kcon,rhcon
      read (2,702) rholio,rholin,rholih,emgpf,emcz,taucz
c
c**********     read in reaction constants     **********
c
      read (2,702) qco1,qco2,qcn,qcw
      read (2,702) rcmbo1,rcmbo2,rcmbn,rcmbw,rcmbh2
      read (2,702) tmelt,tvap,qvap,percen
      read (2,702) conf1,conf2,c2fac
      rcmbo=((100.-percen)*rcmbo1+percen*rcmbo2)/100.
      qco=((100.-percen)*rcmbo1*qco1+percen*rcmbo2*qco2)/(rcmbo*100.)
c
c**********     read in heat transfer correlation coefficients      *********
c
      read (2,702) hin,hingsp,hingss,hinps,hinsam,hinfam
      read (2,702) hinfgs,hinfsg
c
c**********     read in spill parameters     **********
c
      read (2,702) asli,spill,spray,fra,ra
      zli=spill/rhli/asli
c
c**********     read in initial conditions     **********
c
      read (2,702) tczi,tgpzer,tspzer,tsfpi,ta,tlii
      read (2,702) papzer,wo2p,hum,wap,wcp
c
c**********     read in integration control parameters     **********
c
      read (2,705) imeth,dtmin,timef,relerr,delout,output
  705 format(i4,5f12.4)
c
c**********     add needed physical constants & parameters   ********
c**********     for Lithium-CO2 reactions                    ********
c
      rholc3=131.6
      rholc2=102.9
      rhocar=124.7
      qcc=19288.
      ql2c2=7139.
      rcmbco=.22
      rcmbc2=.5783
      rcmbcs=.2313
      rcmca1=2.3133
      rcmca2=2.3133
      rcmcco=0.6309
c
c         ********************************
c**********             options            **********
c         ********************************
c
c
c**********     containment flooding with inert gas option     **********
c
      data tblin,tblout,blowv,exhstv,xblow,wo2b,wn2b,wwab,wcb,xmolab,
     .     cpab,tblow/9*0.0,3*1.0/
c
c**             read in gas flooding parameters if using option        **
c
      if (iblow.ne.1) go to 900
      read (5,702) wo2b,wwab,wn2b,xmolab,cpab,tblow
      read (5,702) blowv,exhstv,tblin,tblout,wcb
c
  900 continue
      wab=1.-wo2b-wn2b-wwab-wcb
c
c**********     emergency space cooling of containment option     **********
c
      data xesc,escr,esctin,escend/4*0.0/
      if (iesc.eq.1) read (5,702) escr,esctin,escend
c
c**********     emergency steel floor liner cooling option     *********
c
      data xsfl,sflcr,sfltin,sflend/4*0.0/
      if (isflc.eq.1) read (5,702) sflcr,sfltin,sflend
c
c**********     aerosol removal from primary containment      **********
c
      if (iarosl .eq. 1) read (5,702) beta
c
c**********     steam injection in steam option   *****************
c
      if (flagst) read (6,702) stmin,stmout,minjr,hinj
      if (flagst.and.flag2) read (6,702) stmin2,stout2,minjr2,hinj2
c
c**********     closure of crack between primary and secondary   *******
c
c
c*****************************************************************
c    print out the input          ***********
c*****************************************************************
c
c
      write (10,800) (name(i),i=1,40)
      write (10,801) iblow,iesc,isflc,iswich,iarosl,flagpn,flag2,
     .               flagsi,flagas,flagc,flagw,flagf,flagisi,
     .               flagpb,flagdf,flagco,flagst
      write (10,802) emconc,cpcon,kcon,rhcon,emli,cpli,akli,rhli,
     .               rhollo,rholin,rholih,emgpf,emcz,taucz,
     .               rholc3,rholc2,rhocar
      write (10,803) vp,chp,cpap,xmola,fra,ra
      write (10,804) tehczp,xmehcp,aehcp,cpehcp,hinecp
      write (10,805) asli,spill,spray,zli
      write (10,2101) conf1,conf2,c2fac
      write (10,806) nl,nl1
      write (10,807) (l(i),i=1,nl)
      write (10,808) (l1(i),i=1,nl1)
      write (10,809) thwc,thfc,gap,kgap,kleak
      write (10,810) estlwp,cpswp,kstlwp,rhswp,awp,thwp
      write (10,811) estlfp,cpsfp,kstlfp,rhsfp,afp,thfp
      write (10,812) hin,hinsam,hingsp,hingss,hinps,hinfam,hinfgs,
     .               hinfsg
      write (10,813) qco,rcmbo,tvap,rcmbh2,percen,qco1,qco2,rcmbo1,
     .               rcmbo2,qcn,rcmbn,tmelt,qcw,rcmbw,qvap,
     .               qcc,ql2c2,rcmbco,rcmbc2,rcmca1,rcmca2,rcmbcs,rcmcco
      write (10,814) tgpzer,tspzer,tczi,tlii,tsfpi,
     .               ta,wo2p,wap,hum,papzer,wcp
      write (10,815) imeth,dtmin,timef,relerr,delout,output
c
c**********      the following parameters are associated with the   *******
c                different options and are written only when used
c
      if (iblow.eq.1.or.isflc.eq.1.or.iesc.eq.1) write (10,819) wo2b,
     .         blowv,cpap,wwab,tblout,cpab,wn2b,tblin,exhstv,tblow,
     .         xmolab,sfltin,sflcr,sflend,esctin,escr,escend,wcb
c
      if (iarosl .eq. 1) write (10,820) beta
c
      if (flagst) write (10,821) stmin,stmout,minjr,hinj
c
      if (flagst.and.flag2) write (10,822) stmin2,stout2,minjr2,hinj2
c
c***************************************************************************
c
  800 format (' ',3(20a4,/),/)
  801 format(' options in effect'/1x,17(1h-)//t10,'iblow = ',i4,t25,
     .'iesc =  ',i4,t40,'isflc =  ',i4,t55,'iswich =',i4//t10,'iarosl ='
     .,i4,t25,'flagpn =',l4,t40,'flag2 =  ',l4,t55,'flagsi =',l4//t10,
     .'flagas =',l4,t25,'flagc =  ',l4,t40,'flagw =  ',l4,t55,'flagf = ',
     .l4//t10,'flagisi =',l4,t25,'flagpb =',l4,t40,'flagdf =',l4,
     .t55,'flagco =',l4//t10,'flagst =',l4//)
  802 format(' physical properties'/1x,19(1h-)//t10,'emconc =',f12.4,
     .t35,'cpcon =  ',f12.4,t60,'kcon =  ',f12.4//t10,'rhcon =  ',f12.4,
     .t35,'emli =   ',f12.4,t60,'cpli =  ',f12.4//t10,'akli =   ',f12.4,
     .t35,'rhli =   ',f12.4,t60,'rholio =',f12.4//t10,'rholin =',f12.4,
     .t35,'rholih =',f12.4,t60,'emgpf = ',f12.4//t10,'emcz =   ',f12.4,
     .t35,'taucz = ',f12.4,t60,'rholc3 =',f12.4//t10,'rholc2 =',f12.4,
     .t35,'rhocar =',f12.4//)
  803 format(' inner containment dimensions'/1x,28(1h-)//t10,'vp =     ',
     .f12.2,t35,'chp =    ',f12.4,t60,'cpap =   ',f12.4//t10,'xmola = ',
     .f12.4,t35,'fra =    ',f12.4,t60,'ra =     ',f12.4//)
  804 format(//' extraneous heat capacity node data'/1x,33(1h-)//t10,
     .'tehczp =',f12.4,t35,'xmehcp =',f12.4,t60,'aehcp = ',f12.4//
     .t10,'cpehcp =',f12.4,t35,'hinecp =',f12.4//)
  805 format(' spill parameters'/1x,16(1h-)//t10,'asli =   ',f12.4,t35,
     .'spill = ',f12.4,t60,'spray = ',f12.4//t10,'zli =    ',f12.4//)
  806 format(/,' wall and floor node data'/1x,24(1h-)//t10,'nl =     ',
     .i2,t35,'nl1 =    ',i2//)
  807 format(' thickness of concrete wall nodes'/1x,31(1h-)//t10,
     .10(f5.3),//t10,10(f5.3)//)
  808 format(/,' thickness of concrete floor nodes'/1x,32(1h-)//t10,
     .10(f5.3),//t10,10(f5.3))
  809 format(//,' parameters associated with outermost containment'/1x,
     .48(1h-)//t10,'thwc =  ',f12.4,t35,'thfc =  ',f12.4,t60,
     .'gap =    ',f12.4//t10,'kgap =  ',f12.4,t35,'kleak = ',f12.4//)
  810 format (' primary steel wall data'/1x,23(1h-)//t10,
     .'estlwp =',f12.4,t35,'cpswp = ',f12.4,t60,'kstlwp =',f12.4//t10,
     .'rhswp = ',f12.4,t35,'awp =    ',f12.4,t60,'thwp =  ',f12.4//)
  811 format (' priamry steel floor data'/1x,24(1h-)//t10,
     .'estlfp =',f12.4,t35,'cpsfp = ',f12.4,t60,'kstlfp =',f12.4//t10,
     .'rhsfp = ',f12.4,t35,'afp =   ',f12.4,t60,'thfp =  ',f12.4//)
  812 format(' heat transfer correlation coefficients'/1x,38(1h-)//
     .t10,'hin =   ',f12.4,t35,'hinsam =',f12.4,t60,'hingsp =',f12.4//
     .t10,'hingss =',f12.4,t35,'hinps = ',f12.4,t60,'hinfam =',f12.4//
     .t10,'hinfgs =',f12.4,t35,'hinfsg =',f12.4//)
  813 format(' combustion parameters'/1x,21(1h-)//t10,'qco =   ',f12.4,
     .t35,'rcmbo = ',f12.4,t60,'tvap =  ',f12.4//t10,'rcmbh2 =',f12.4,
     .t35,'percen =',f12.4,t60,'qco1 =  ',f12.4//t10,'qco2 =  ',f12.4,
     .t35,'rcmbo1 =',f12.4,t60,'rcmbo2 =',f12.4//t10,'qcn =   ',f12.4,
     .t35,'rcmbn = ',f12.4,t60,'tmelt = ',f12.4//t10,'qcw =   ',f12.4,
     .t35,'rcmbw = ',f12.4,t60,'qvap =  ',f12.4//t10,'qcc =   ',f12.4,
     .t35,'ql2c2 = ',f12.4,t60,'rcmbco =',f12.4//t10,'rcmbc2 =',f12.4,
     .t35,'rcmca1 =',f12.4,t60,'rcmca2 =',f12.4//t10,'rcmbcs =',f12.4,
     .t35,'rcmcco =',f12.4//)
  814 format (' initial conditions'/1x,18(1h-)/5x,'primary'//
     .t10,'tgpzer =',f12.4,t35,'tspzer =',f12.4,t60,'tczi =  ',f12.4//
     .t10,'tlii =  ',f12.4,t35,'tsfpi = ',f12.4,t60,'ta =    ',f12.4//
     .t10,'wo2p =  ',f12.4,t35,'wap =   ',f12.4,t60,'hum =  ',f12.4//
     .t10,'papzer =',f12.4,t35,'wcp =   ',f12.4//)
  815 format(' integration control parameters'/1x,30(1h-)//t10,
     .'imeth = ',i4,t35,'dtmin = ',f12.4,t60,'timef = ',f12.4//t10,
     .'relerr =',f12.4,t35,'delout =',f12.4,t60,'output =',f12.4//)
  819 format(' miscellaneous input associated with various options'/1x
     .51(1h-)/5x,'inert gas flooding'//t10,'wo2b =  ',f12.4,t35,
     .'blowv = ',f12.4,t60,
     .'cpap =  ',f12.4//t10,'wwab =  ',f12.4,t35,'tblout =',f12.4,t60,
     .'cpab =  ',f12.4//t10,
     .'wn2b =  ',f12.4,t35,'tblin = ',f12.4,t60,'exhstv =',f12.4//t10,
     .'tblow = ',f12.4,t35,'xmolab =',f12.4//5x,
     .'steel floor cooling'//t10,'sfltin =',f12.4,t35,'sflcr = ',f12.4,
     .t60,'sflend =',f12.4//5x,'emergency space cooling'//t10,
     .'esctin =',f12.4,t35,'escr =  ',f12.4,t60,'escend =',f12.4//5x,
     .t10,'wcb =   ',f12.4//)
  820 format (' aerosol removal from primary containment'/1x,41(1h-)//
     .t10,'beta =  ',f12.4//)
  821 format (' steam injection to containment'/1x,31(1h-)//t10,
     .'stmin =',f12.4,t35,'stmout =',f12.4,t60,'minjr =',f12.4//5x,
     .t10,'hinj =',f12.4//)
  822 format (' steam injection to secondary'/1x,29(1h-)//t10,
     .'stmin2 =',f12.4,t35,'stout2 =',f12.4,t60,'minjr2 =',f12.4//5x,
     .t10,'hinj2 =',f12.4//)
 2101 format (' combustion branching factor or ratio'/1x,37(1h-)//
     .t10,'conf1 = ',f12.4,t35,'conf2 = ',f12.4,t60,'c2fac = ',f12.4//)
c**********************************************************************
c                        ********************                         *
c************************     options      *****************          *
c                        ********************                         *
c    see litfire users guide for dimensions of option variables       *
c                                                                     *
c    in this step the secondary cell, pan geometry, and concrete wall *
c        and floor variables are read in  and written                 *
c**********************************************************************
      n=1
      num=0
      if (flagpb .and. spray .gt. 0.) go to 984
      if (flagc .and.flagpn) go to 980
      if (flagco .and. flagas) go to 2001
      if (flagco .and. flagpb) go to 2003
      if (flagco .and. wo2p .ne. 0.) go to 2005
      flagn=.true.
      if (flag2) call cell2
      if (flagpn) call pan
      if (flagas) call injec
      if (flagc) call concc
      if (flagpb) call lipb
      if (flagdf) call lidiff
      if (flagst) call table
      flagn=.false.
      if (flagisi) call si
c
c*********************************************************************
c    initialize program variables           ***********
c*********************************************************************
c
      flagl=.false.
c
      icz=1
      icmb=1
      ilit=1
      icni=1
      ico2i=0
c
      timeo=-.001
      tau=120.
c  tau should be time dependent see note by mst.
      sigma=4.7619e-13
      gin=32.2
      ipage=40
      delt=dtmin
c
c   call sub. CO2 if CO2 atmosphere option is used.
c
      if (flagco) call co2(icmb,icni,ico2i,n)
      if (flagco) ico2i=1
c
c**********   initialize primary containment variables   **********
c
      data cmbro,cmbrn,cmbrw,cmbrhi,dfilm,hf,hb,libp,lilox,lilni,leako,
     .     mlinip,mlinp,mlihp,mh2p,oxlb,oxlbi,outint,roxlb,rnilb,rwalb,
     .     cmbrco,cmbrc2,lilca,lilc2,lilcar,mco2ip,mlc3p,mlc2p,rcolb,
     .     rc2lb,time,zz1,zz2,zz4,zz5,zz6,zz7,zz8,zz9,zzep,
     .     ul,ulz,uv,uvz,mwl,mwlz,mwv,mwvz,fpg,fpw/49*0.0,2*1.0/
c
      fmleft=1.0
      lis=spill*spray
      lit=spill-lis
      lilp=lit
      lil=lilp
      wn2p=1.-wo2p-wap-wcp
      xmolp=wo2p*32.+wn2p*28.+wap*xmola+wcp*44.
      rinp=1545./xmolp
      rair=rinp
      tehcp=tehczp
      tli=tlii
      tcz=tczi
      tsp=tspzer
      tsfp=tsfpi
c************** initialize the amount of water vapor in containment ***
      flagn=.true.
      if (flagst) call steam
      flagn=.false.
c ***********************************
      rhoalp=papzer*144./rinp/tgpzer
      if(flagst) rhoaip=(papzer-ph2o)*144./rinp/tgpzer
      rhoap=rhoaip
      mniip=wn2p*rhoaip*vp
      moxip=wo2p*rhoaip*vp
      mnip=mniip
      mco2ip=wcp*rhoaip*vp
      mco2p=mco2ip
      mlioip=lis*(1.+rcmbo)/rcmbo
      if (ico2i .eq. 1) mlioip=0.
      if (ico2i .eq. 1) mlc3ip=lis*(1.+rcmbcs)/rcmbcs
      maip=wap*rhoaip*vp
      map=maip
      if(flagst) wwap=mwv/(mnip+moxip+map+mwv)
c
c**********   initialize option variables  ********
c
      if (flag2) call cell2
      if (.not. flag2) rbreak=0.0
      if (flagpn) call pan
      if (flagc) call concc
      flagn=.true.
      if (flagw) call concw
      if (flagf) call concf
      flagn=.false.
      blowr=1.35e-03*blowv
      exhstr=1.35e-03*exhstv
      stick=0.0
      if (iarosl .eq. 1) stick=awp/(vp*beta)/12.
      if (stick .ge. 1.0) go to 986
      if (stick .gt. .25) write (11,823)
  823 format (' aerosol removal fraction is greater than one quarter
     .  of aerosol'/'inventory.  time step has been decreased to insure
     .  stability.')
      if (stick .gt. .25) ipage=ipage+2
c
c**********    conversion to ft. - lb. - sec.     **********
c
      akli=akli/3600.
      kstlwp=kstlwp/3600.
      kstlfp=kstlfp/3600.
      kcon=kcon/3600.
      kgap=kgap/3600.
c
c
c**************************************
c*   spray fire computation started    *
c**************************************
c
c
c**  check that enough oxygen or CO2 is left for pool fire after spray fire  ***
c
      if (ico2i .ne. 1) then
      oxlfs=wo2p*rhoap*vp-lis/rcmbo
       if (oxlfs .lt. 0.0)then
       lis=rcmbo*wo2p*rhoaip*vp
       oxlfs=0.0
       endif
      else
      co2lfs=wcp*rhoap*vp-lis/rcmbco
       if (co2lfs .lt. 0.0)then
       lis=rcmbco*wcp*rhoaip*vp
       co2lfs=0.0
       endif
      end if
c
```

```
      if (lis.le.0.0) go to 902
      to=tgpzer
      if (ico2i .ne. 1)then
      qin=lis*(qco+cpli*(tli-to))
      else
      qin=lis*(qcc+cpli*(tli-to))
      endif
      ff2=qin
      te=tgpzer+1.
  901 continue
c**********              specific heat              ****************
c    cp = .0602*t**.326  t = deg. r   for lithium oxide             *
c    cp = .761   for lithium carbonate
c    if a different reaction product is desired, the integral of the *
c    desired product must be substituted in qout1.                  *
c*******************************************************************
      cplc3p=.751
      cpco2p=.201
      if (ico2i .ne. 1) then
      qout1=(1.+rcmbo)/rcmbo*lis*(0.0602/1.326)*(te**1.326-to**1.326)
      qout3=oxlfs*(.184*(te-to)+3.2e-6/2.*(te**2.-to**2.)+1.36e04*
     .          (1./te-1./to))
      qout5=wcp*rhoap*vp*cpco2p*(te-to)
      else
      qout1=(1.+rcmbcs)/rcmbcs*lis*cplc3p*(te-to)
      qout3=co2lfs*cpco2p*(te-to)
      end if
      qout2=wn2p*rhoap*vp*(.172*(te-to)+8.57e-06/2.*(te*te-to*to)+
     .   1.02e-09/3.*(te**3.-to**3.))
      qout4=mwv*(0.44*(te-to))+wap*rhoap*vp*cpap*(te-to)
      ff1=qin-qout1-qout2-qout3-qout4
      if (ico2i .ne. 1) ff1=ff1-qout5
      if (ff1*ff2.lt.0.) go to 903
      te=te+1.
      if (te.gt.1.0e06) go to 979
      ff2=ff1
      go to 901
c*****   portion of program for getting initial gas temp. and press. ***
  902 continue
      te=tgpzer
  903 continue
      tgp=te
      moxp=moxip-lis/rcmbo
      if (ico2i .eq. 1) moxp=0.0
      moxip=moxp
      if (ico2i .eq. 1) mco2p=mco2ip-lis/rcmbco
      mliop=mlioip
      xmairp=mnip/28.+moxp/32.+map/xmola+mco2p/44.
      pzerop=1545.*xmairp*tgp/144./vp
      if(flagst) pzerop= pzerop+ph2o
      pap=pzerop
      tgpzer=tgp
c
c ***determine the initial energy of the vapor region***
      if(flagst)call steam
c ***********************************
      write (10,825) tgp,pzerop
  825 format (//' spray fire results'/1x,18(1h-)//5x,'tgpzer = ',f6.1,
     . '     pzerop = ',f8.3///)
      qqq=20.
c**********     spray fire computation concluded     **********
c
      call init
```

```
c
c*************************************
c*   start of dynamic cycle          *
c*   ______________________          *
c*   start of integration cycle      *
c*************************************
c
  200 continue
c
c ****************************************************
c ************* reset rates of change to zero *************
c
      zz1=0.0
      zz2=0.0
      zz4=0.0
      zz5=0.0
      zz6=0.0
      zz7=0.0
      zz8=0.0
      zz9=0.0
      zzep=0.0
      ulz=0.0
      uvz=0.0
      mwlz=0.0
      mwvz=0.0
      ulz2=0.0
      uvz2=0.0
      mwlz2=0.0
      mwvz2=0.0
c
c*****   injection of gases to model hedl experiment  *****
      moxinj=0.0
      mniinj=0.0
      if (flagas) call injec
c
c*****     compute physical properties dependent on temperature   *****
c*****  calculate air composition and specific heat at const. volume   ******
c
      mairp=moxp+mnip+mh2p+map+mco2p
      xmolp=(28*mnip+32*moxp+2*mh2p+xmola*map+44*mco2p)/mairp
      rair=1545./xmolp
      rhoap=mairp/vp
      foxp=moxp/mairp
      fwap=mwv/(mwv+mairp)
      fnip=mnip/mairp
      fwcp=mco2p/mairp
      cpo2p=(0.184+3.2e-06*tgp-1.36e04/(tgp*tgp))
      cpmoxp=cpo2p*moxp
      cpn2p=(0.172+8.57e-06*tgp+1.02e-09*tgp*tgp)
      cpmnip=cpn2p*mnip
      cpwa=0.44
      cph2=2.48
      cpco2p=0.193
      cpmcop=cpco2p*mco2p
      cplih=0.67
      cpliop=0.0602*tgp**.326
      cplinp=0.3368+3.67e-04*tgp
      cpmlop=cpliop*mliop
      cplc3p=0.761
      cpmlcp=cplc3p*mlc3ip
      cplc2p=0.7
      cpcarp=0.17
c
```

```
c
      rhli=33.49-.0035*(tli-460.)
      akli=(10.48+2.767e-03*(tli-817.)-0.322e-06*(tli-817.)**2)/1488.
      cpfac=0.004938*tli-6.20741
      cpli=1.0037-.01063*cpfac+.00564*cpfac**2-.001279*cpfac**3
      cpli=((lit-libp)*cpli+lilox*cpliop+lilni*cplinp+lilca*cplc3p+
     .     lilcar*cpcarp+lilc2*cplc2p)/lilp
c
c  The following statements are made in order to simulate HEDL LC-2
c  experiment in which the lithium was transferred into the pan
c  during 0 to 150 sec period.
c  In addition, the following statements will not be used in normal
c  cases; the first column of these statement will be filled with
c  "c" which make them inactive.  If you want to try this option,
c  delete these "c"'s and use appropriate input files (see T.K. Gil
c  master thesis).  And make sure to insert the "c"'s again for
c  normal case runs.
c
c      if (qqq .ge. 150.) goto 11
c      if (time .ge. qqq) then
c      lit=lit+0.14667
c      qqq=qqq+1.
c      rhlix=33.39-.0035*(tli-460.)
c      aklix=(10.48+2.767e-03*(tlii-817.)-0.322e-06*(tlii-817.)**2)/
c     .      1488.
c      cpfacx=0.004938*tlii-6.20741
c      cplix=1.0037-.01063*cpfacx+.00564*cpfacx**2-.001279*cpfacx**3
c      cplix=0.14667*cplix
c      cpli=cpli+cplix
c      rhli=rhli*(lit-0.14667)/(lit+.14667)+rhlix*0.14667*(lit+
c     .     0.14667)
c      akli=akli*(lit-0.14667)/(lit+.14667)+aklix*0.14667*(lit+
c     .     0.14667)
c      else
c      continue
c      end if
c 11   continue
c  up to the above line is used for LC-2 lithium tansfer simulation
c
c **************************************************************
c
      if (flagpb) call lipb
c
c*****  two millimeters are assumed to cover the pool optically  *****
      zp=(lilox/rholio+lilni/rholin+lilca/rholc3+lilc2/rholc2+
     .    lilcar/rhocar)/asli
      emf=0.9
      if (emli.lt.emf)emli=0.2+(emf-0.2)*zp/0.00656
c
      htcpgp=cpmoxp+cpmnip+cpmlop+cpap*map+cplinp*mlinp+cplih*mlihp+
     .   cph2*mh2p+cpmcop+cpmlcp
c
      cpa=htcpgp/(mairp+mlinp+mlihp+mlc3ip)
c
      emgp=1.-exp(-(mliop/rholio+mlinp/rholin+mlihp/rholih+mlc3p/
     .    rholc3)*2.27e05*chp/vp/ra)
      emgp=emgp*emgpf
      if(emgp.le.0.005)emgp=0.005
c************** calculate the emissivity of the air-vapor mixture ***
c************** and determine the primary air-vapor mixture temperature
c
      if(flagst) call steam
c
```

```
c***** calculating radiative interchange factors   *****
c   fpg and fpw represent view factors from the pool. they are
c   initialized as unity if pan is not presen. initialized in
c   pan option if it is used.   taucz is used instead of (1.-emcz)
c   to more flexibly model combustion zone-pool coupling.
c
      rifpw=1./((1.-emli)/emli+(1.-estlwp)*asli/estlwp/awp+1./
     .   ((1.-emgp)*(icz*(taucz-1.)+1.)*fpw+emgp/(asli/awp+1./
     .   fpg/(icz*(taucz-1.)+1.))))
      rifczw=1./((1.-emcz)/emcz+(1.-estlwp)*asli/estlwp/awp+1./
     .   ((1.-emgp)+emgp/(1.+asli/awp)))
      rifpg=(emli*emgp)/((1.-emli)*emgp+emli/fpg/(icz*(taucz-1.)+1.))
      rifczg=(emcz*emgp)/((1.-emcz)*emgp+emcz)
      rifscw=(estlwp*emconc)/(estlwp+emconc-estlwp*emconc)
      rifscf=(estlfp*emconc)/(estlfp+emconc-estlfp*emconc)
      rifczp=(emli*emcz)/(emcz+emli-emcz*emli)
c
c**********     calculating gas convection coefficient     **********
c
c the following calculation invokes reynold's analogy between
c heat and mass transfer by assuming that
c
c                         1/3
c               sh=c(grsc)
c
c the sherwood number, (h  l / d), is defined by the relation:
c                        m
c
c            j = h  rho (w  -w )
c                 m       1   2
c
c reynold's analogy, together wih the lewis relation, gives us:
c
c                     h  = h  /rho c
c                      m    c       p
c
c in litfire, w  is assumed to be zero.
c              2
c
c  pool or combustion zone to primary gas
      if (icz .eq. 1)  t1=0.5*(tgp+tcz)
      if (icz .eq. 0) t1=0.5*(tgp+tli)
      b1= 1.0/t1
      d1=((4.94e-05*t1+0.0188)/(rhoap*3600.))**2
      ak1=(0.014+1.92e-05*(t1-460.))/3600.
      diff=241.57/(132.0+t1/1.8)*(t1/493.2)**2.5/3600.
c
c  the following statements may have to be changed when code
c  runs can be compared to actual experimental results to determine
c  the effect of steam on the gas diffusion to the combustion zone
c
c   account for steam effects on diffusion
      if(flagst)then
      vst=vvg
      va=1./rhoap
      rhotot=rhoap+1./vg
      muvcz=(11.4/((t1/1.8)**2-884*t1/1.8+1.36e06))/1.488
      temp2=(sqrt(d1)*rhoap/muvcz)**.5
      phia=.218942*(1.+0.88808*temp2)**2
      phiw= .277605*(1.+1.2603/temp2)**2
      stmfac= vst/(vst+va*phia)
      airfac= va/(va+vst*phiw)
      mudiff= airfac*muvcz+stmfac*(sqrt(d1)*rhoap)
      d1= (mudiff/rhotot)**2
      akvap1=muvcz*cpwv
      ak1= stmfac*ak1+airfac*akvap1
      endif
c ****************************************
      if (icz .eq. 1) exx=(gin*b1*abs(tcz-tgp)/d1)
      if (icz .eq. 0) exx=(gin*b1*abs(tli-tgp)/d1)
      if   (exx .le. 0.0) go to 985
      ex1 = (exx)**0.3333
      hfinf=hin*diff*ex1
      hbinf=hin*ak1*ex1
      if (tau .lt. delt) tau=delt
      hf=hf+(hfinf-hf)*delt/tau
      hb=hb+(hbinf-hb)*delt/tau
c
c******* check for boiling or condensation if water is present *******
c******* and calculate gas heat transfer coefficients ***************
      if(flagst)call steam
c
c**** without steam, calculate the coefficients normally ****
c
c  primary gas to primary steel liner
       hgwp=hingsp*akexx(tgp,tsp,rhoap)
c  primary gas to primary extraneous heat capacity
       hehcp=hinecp*akexx(tgp,tehcp,rhoap)
c ***calculate heat transfer coefficients with steam present***
      if(flagst) call steam
c **********************************************
c  primary steel liner to ambient if not two cell or concrete option
      if (.not. (flag2 .or. flagw)) ha=hinsam*akexx(tsp,ta,.074)
c  primary steel floor to ambient (if not two cell or concrete)
      if (.not. (flag2 .or. flagf)) hamf=hinfam*akexx(tsfp,ta,.074)
c   *****     calculating thermal diffusivities between nodes     *****
      if (flagw) call concw
      if (flagf) call concf
      cehcgp=hehcp*aehcp/htcpgp
      cgpehc=hehcp*aehcp/xmehcp/cpehcp
      c1=kstlwp*hgwp*awp/htcpgp/(thwp*hgwp/2.+kstlwp)
      c6=kstlwp*hgwp/(rhswp*cpswp*thwp*(thwp*hgwp/2.+kstlwp))
c  the next thermal diffusivity is valid only if no wall concrete and
c  no secondary containment cell, and is between steel liner and ambient
      if (.not. (flagw.or.flag2)) c11=kstlwp*ha/(rhswp*cpswp*thwp*
     .   (kstlwp+thwp*ha/2.))
      if (.not. (flagw.or.flag2)) c12=kstlfp*hamf/(rhsfp*cpsfp*thfp*
     .   (kstlfp+thfp*hamf/2.))
c
c
c*********************************************************************
c   repeat above calculations dependent on temperature for secondary   *
c                    containment                                       *
      if (flag2) call cell2
c*********************************************************************
c
c***** testing to see if emergency space cooling or steel cooling in effect
      if (time .gt. esctin) xesc=1.
      if (time .gt. escend) xesc=0.
      if (time .gt. sfltin) xsfl=1.
      if (time .gt. sflend) xsfl=0.
c
c*******  steam injection option ****************
c
      xinj=0.0
      if((flagst).and.(time .gt. stmin).and.(time.lt.stmout))xinj=1.0
```

```
c
c*****     lithium lead diffusion calculation in preperation     *****
c                for combustion rate calculation
      if (flagdf) call lidiff
c***************       testing for combustion     *****************
      icni=0
      tez=(tcz+tli)/2.
      if (tez .le. 2520. .and.foxp.le.0.28 .and. mnip.gt.0.0) icni=1
      if (.not.(ilit .eq. 0 .or. ico2i .eq. 0)) go to 909
      if (.not.(ilit.eq.0 .or.(icmb.eq.0 .and. icni.eq.0) .or. tli.lt.
     .    tmelt)) go to 909
      if (icz.eq.1)write (11,827)icz,icni,ilit,icmb,tcz,foxp,tli,time
  827 format(' combustion has just stopped.  parameters are  icz=',i1,
     .'  icni=',i1/' ilit=',i1,'  icmb=',i1,'  tcz= ',f8.2,'  foxp= ',
     .   f7.3,'  tli= ',f8.2,'   at time= ',f9.2)
      if (icz.eq.1) ipage=ipage+2
      go to 910
c
c
c*****************************************************************
c      computations using combustion zone model       ***********
c*****************************************************************
c
c**********     computing rate of lithium combustion     **********
  909 rn2=0.
      ro2=0.
      icz=1
      if (ico2i .eq. 1) go to 22
      cmbrn=hf*fnip*rhoap*rcmbn
      if (rr(tli) .le. cmbrn) cmbrn=rr(tli)
      if (tez .le. 2520.) rn2=1-(foxp/(foxp+fnip))**0.18
      if (tez .gt. 2520. .or. foxp .gt. 0.28) rn2=0.0
      ro2=(1-fnip/(foxp+fnip))**0.02
c
c       rc2 is the inhibition factor for the lithium-CO2 reaction rate
c       which is strongly depending on the degree of carbonate layer
c       buildup.  Here, 1 inch of carbonate is assumed to stop the
c       reaction completely when the lithium pool temp. is below 400 C.
c
   22 if (tli .le. 1000.) rc2=0.0
      if (lilca .le. 0. .or. tli .gt. 1180) rc2=1.
      ratio1=lilca/rholc3/asli
      if (ratio1 .gt. 0.08333 .and. tli .le. 1180.) rc2=0.
      if ((ratio1 .le. 0.08333 .and. ratio1 .gt. 0.) .and. tli .le.
     .          1180.) rc2=(1-ratio1/.08333)**1.5
      cmbro1=hf*foxp*rhoap*rcmbo*ro2
      cmbro2=hf*fwcp*rhoap*rcmcco*conf1
      cmbro=cmbro1+cmbro2
      cmbrn=cmbrn*rn2
      if(flagst)cmbrw=hf/vg*rcmbw
      cmbrco=rc2*hf*fwcp*rhoap*rcmbco*conf2
      cmbrc2=c2fac*(cmbrco+cmbro2)*rcmbc2
   33 if (ico2i .eq. 1 .and. tli .gt. 1180.) cmbro=hf*fwcp*rhoap*
     .    rcmcco*conf1
      if (ico2i .eq. 1 .and. tli .gt. 1180.) cmbrco=hf*fwcp*rhoap*
     .    rcmbco*conf2
      if (ico2i .eq. 1) cmbrc2=c2fac*(cmbrco+cmbro)
      cmbr = cmbro + cmbrn + cmbrw + cmbrco + cmbrc2
      if (.not. flagdf) go to 1909
      if (cmbr .lt. xlidot) go to 1909
      cmbro=cmbro*xlidot/cmbr
      cmbrn=cmbrn*xlidot/cmbr
      cmbrw=cmbrw*xlidot/cmbr
      cmbrco=cmbrco*rlidot/cmbr
      cmbrc2=cmbrc2*xlidot/cmbr
      cmbr=cmbro+cmbrn+cmbrw+cmbrco+cmbrc2
 1909 continue
      if (cmbr*3600. .lt. 0.2) go to 910
      rnilb=cmbrn*asli/rcmbn
      roxlb=cmbro*asli/rcmbo
      rwalb=cmbrw*asli/rcmbw
      rcolb=cmbrco*asli/rcmbco
      rc2lb=cmbrc2*asli/rcmbc2
c
c**********     computation of lithium vapor diffusion     **********
      tfeff=0.002*(tcz+tli)/2.-3.92
      pliv=(10.**(4.8831-14180.2/tli))*14.
      if (flagpb) pliv=actvty(xalloy)*pliv
      rholiv=pliv*144./rinp/tli
      diffli=3.56e-03*((tli/460.)**1.81)/pap
      dfilm=diffli*rholiv/cmbr
      efilm=dfilm*12.
c
      knit=.0432+tfeff*(.0078-tfeff*(8.2e-04+tfeff*2.08e-04))
      klit=0.55+tfeff*(-4.99e-04+tfeff*1.206e-07)
      kfilm=(pliv*(klit-knit)+pap*knit)/14.7
c**********     computation of heat transfer coefficients     **********
      yapcz=kfilm*akli*asli/(dfilm*akli+kfilm*zli/2.)
c***** this heat capacity is sheer guess work the 0.1 is for low comb. rates
      cpmcz=asli*((1.+rcmbo)/rcmbo*cmbro*cpliop+(1.+rcmbn)/rcmbn*cmbrn*
     .   cplinp+(1+rcmbcs)/rcmbcs*cmbrco*cplc3p+(1.+rcmbc2)/rcmbc2*
     .   cmbrc2*cplc2p+
     .   lilcar*cpcarp+((1.+rcmbw)/rcmbw-(1./rcmbh2))*cmbrw*cpwv+
     .   (1.+rcmbh2)/rcmbh2*cmbrw*cph2+rn2*hf*fnip*rhoap*cpn2p)*300.+1.
c*****   in this case of Li-CO2 reaction, cpmcz may vary.       ********
c*****   the following is a sheer guess work.                   ********
      if (ico2i .eq. 1) cpmcz=(cpmcz-1.)/300.*100.
      if (cpmcz/asli .le. 0.001) cpmcz=0.001*asli
      cgcz=hb*asli/cpmcz
      cczg=hb*asli/htcpgp
      cpcz=yapcz/cpmcz
      cczp=yapcz/(cpli*lil)
      ccz=(cmbro*qco+cmbrn*qcn+cmbrw*qcw+cmbrco*qcc+cmbrc2*ql2c2)*asli
      if (ico2i .eq. 1 .and. tli .gt. 1180.) ccz=(cmbro*qco+cmbrco*qcc+
     .    cmbrc2*ql2c2)*asli
      clist=2.*asli*akli*kstlfp/(lil*cpli*(zli*kstlfp+thfp*akli))
      csbli=2.*asli*akli*kstlfp/(rhsfp*asli*thfp*cpsfp*(zli*kstlfp+
     .    thfp*akli))
      qradp=sigma*asli*(tcz**4-tli**4)*rifczp
      qradw=sigma*asli*(tcz**4-tsp**4)*rifczw
      qradg=sigma*asli*(tcz**4-tgp**4)*rifczg
      rczw=qradw/(thwp*awp*rhswp*cpswp)
      rczp=qradp/(lil*cpli)
      rczg=qradg/htcpgp
      qrady=sigma*asli*(tli**4-tsp**4)*rifpw
      qradz=sigma*asli*(tli**4-tgp**4)*rifpg
      rliw=qrady/(thwp*awp*rhswp*cpswp)
      rwli=qrady/cpli/lil
      rgli=qradz/cpli/lil
      rlig=qradz/htcpgp
c
c
c**************************************************************
c*   calculating temperature rates of change  with combustion  *
c**************************************************************
c
```

```
c**********     calculate comb. zone temp. rate of change  deg.r/sec.      ******
      zz6=(ccz-(qradp+qradw+qradg))/cpmcz+qvap*cmbr*asli/cpmcz
     .        -cpcz*(tcz-tli)-cgcz*(tcz-tgp)
c
c**********     calc. lithium temp. rate of change  deg. r/sec.     **********
      zz1=cczp*(tcz-tli)+rczp-clist*(tli-tsfp)-qvap*cmbr*asli*cczp/yapcz
     .      -rwli-rgli
c
c**********     calc. cell gas temp. rate of change  deg. r/sec.     **********
      if(.not.flagst)zz4=c1*(tsp-tgp)+cczg*(tcz-tgp)+rczg+rbreak+xblow*
     .              blowr*cpab*(tblow-tgp)/htcpgp-escr*xesc/htcpgp+
     .              cehcgp*(tehcp-tgp) + rlig
c
c
c**********     calc. wall steel temp. rate of change  deg. r/sec.     *********
      if(mwv.le.0.0)zz5=zz5+c6*(tgp-tsp)+rczw+rliw
c
c************* computations with steam present *********
c
c vapor region energy rate of change
      if(flagst)uvz=uvz+cczg*htcpgp*(tcz-tgp)+qradg+qradz+
     .          xblow*blowr*cpab*tblow-xesc*escr
      if(flagst.and. mwv.le.0.)uvz=uvz+c1*(tsp-tgp)*htcpgp+
     .   cehcgp*(tehcp-tgp)*htcpgp
c wall liner temp rate of change
      if(mwv.gt.0.0)zz5=zz5+rczw+rliw
c
      go to 911
c
c
c*************************************************
c*   computations without combustion zone model     *
c*************************************************
c
  910 continue
      icz=0
      cmbr=0.0
      rn2=0.0
      yalig=akli*hb*asli/(akli+hb*zli/2.)
      clig=yalig/htcpgp
      qradw=sigma*asli*(tli**4-tsp**4)*rifpw
      qradg=sigma*asli*(tli**4-tgp**4)*rifpg
      rliw=qradw/(thwp*awp*rhswp*cpswp)
      rwli=qradw/cpli/lil
      rgli=qradg/cpli/lil
      rlig=qradg/htcpgp
      cgli=yalig/(lil*cpli)
      clist=2.*asli*akli*kstlfp/(lil*cpli*(zli*kstlfp+thfp*akli))
      csbli=2.*asli*akli*kstlfp/(rhsfp*asli*thfp*cpsfp*(zli*kstlfp+
     .     thfp*akli))
c
c*********************************************
c*   calculating temperature rates of change   *
c*********************************************
c
c**********     calc. lithium temp. rate of change  deg. r/sec.     **********
      zz1=cgli*(tgp-tli)-clist*(tli-tsfp)-rwli-rgli
c  let combustion follow pool temperature for possible reignition
      zz6=(tli-tcz)/delt
c
c  ******* calc. cell gas temp. rate of change deg . r/sec.    ********
      if (.not.flagst) zz4= c1*(tsp-tgp) +clig*(tli-tgp) +
     .                rlig +rbreak +xblow*blowr*cpab*(tblow-tgp)/
     .               htcpgp -escr*xesc/htcpgp +cehcgp*(tehcp-tgp)
c
c
c  with steam present — vapor region energy rate of change
c
      if(flagst)uvz=uvz+yalig*(tli-tgp)+qradz+xblow*blowr*cpab*tblow
     .           -xesc*escr
      if(flagst.and. mwv.le.0.0)uvz=uvz+c1*(tsp-tgp)*htcpgp+
     .   cehcgp*(tehcp-tgp)*htcpgp
c
c**********     calc. wall steel temp. rate of change  deg. r/sec.     *********
      if(mwv.le.0.0)zz5=zz5+c6*(tgp-tsp)+rliw
c  with steam present—
      if(mwv.gt.0.0)zz5=zz5+rliw
  911 continue
c
c**********************************************
c*   computations valid with either model      *
c**********************************************
c
      if(mwv.le.0.0)zzep=cgpehc*(tgp-tehcp)
c**********     calc. floor steel temp. rate of change  deg. r/sec.  ********
      zz7=-xsfl*sflcr*12./(thfp*afp*rhsfp*cpsfp)
c
      if (flag2) go to 915
      if (.not. flagw) qradc=sigma*awp*(tsp**4-ta**4)*estlwp
      if (flagw) qradc=sigma*awp*(tsp**4-tc(1)**4)*rifscw
      radc=qradc/(thwp*awp*rhswp*cpswp)
      if (.not. flagw) zz5=zz5-c11*(tsp-ta)-radc
      if (flagw) zz5=zz5-c7*(tsp-tc(1))-radc
      if (.not. flagf) qradb=sigma*asli*(tsfp**4-ta**4)*estlfp
      if (flagf) qradb=sigma*asli*(tsfp**4-tb(1)**4)*rifscf
      radb=qradb/(thfp*asli*rhsfp*cpsfp)
      if (.not. flagf)  zz7=zz7+csbli*(tli-tsfp)-c12*(tsfp-ta)-radb
      if (flagf) zz7=zz7+csbli*(tli-tsfp)-c8*(tsfp-tb(1))-radb
  915 continue
      if (flag2) call cell2
      if (flagf) call concf
      if (flagw) call concw
      if (flagpb) call lipb
      if (flagdf) call lidiff
c
c
c**********calculations with suspended pan geometry*********
c
      if (flagpn) call pan
c
c******* calculations using combustion of concrete (breach of steel liner)****
c
      if (flagc) call concc
c
c*********************************
c**    calculating overpressure   **
c*********************************
      xmairp=moxp/32.+mnip/28.+map/xmola+mco2p/44.+mh2p/2.
      pap=1545.*xmairp*tgp/144./vp
      if(flagst)pap=pap+ph2o
      overpp=pap-papzer
      if (time.gt.tblin) xblow=1.
      if (time.gt.tblout) xblow=0.
c
c*********************************
c**    calcu. total  leakage    ***
```

```
c***********************************
c
      leak=0.0
      if (flag2) call cell2
      if (flag2) go to 932
      if (pap .gt. 14.7) leak=kleak*(pap-14.7)**0.5
      xmdot=0.0
      fouts=0.0
      foutp=exhstr/mairp*xblow+leak
  932 continue
      fmleft= exp(-outint)
      fmleak=1.-fmleft
c
c **** calculate hydrogen ratio (moles H2 to total moles of gas)
      hratio=mh2p/2./(xmairp+mwv/18.)
c
c************************
c*   do integrations    *
c************************
c
      libp=intgrl(0.,cmbr*asli)
      lilox=intgrl(0.,(1.+rcmbo)/rcmbo*cmbro*asli*(1.-fra))
      if (ico2i .eq. 1) lilox=intgrl(0.,(1+.535)*cmbro*asli)
      lilni=intgrl(0.,(1.+rcmbn)/rcmbn*cmbrn*asli*(1.-fra))
      lilca=intgrl(0.,(1.+rcmbcs)/rcmbcs*cmbrco*asli*(1.-fra))
      lilc2=intgrl(0.,(1.+rcmbc2)/rcmbc2*cmbrc2*asli)
      lilcar=intgrl(0.,cmbrco/rcmca2*asli+cmbro2/rcmca1*asli-
     . cmbrc2/rcmbc2*asli)
      if (ico2i .eq. 1 .and. tli .gt. 1180.) lilcar=intgrl(0.,
     . cmbrco/rcmca2*asli+cmbro/rcmca1*asli-cmbrc2/rcmbc2*asli)
      oxlb=intgrl(oxlbi,roxlb)
      if (ico2i .eq. 1) oxlb=0.0
      tcz=intgrl(tczi,zz6)
      tli=intgrl(tlii,zz1)
      if(.not.flagst)tgp=intgrl(tgpzer,zz4)
      tsp=intgrl(tspzer,zz5)
      tehcp=intgrl(tehczp,zzep)
      tsfp=intgrl(tsfpi,zz7)
      if (ico2i .eq. 1) then
      moxp=0.0
      else
      moxp=intgrl(moxip,wo2b*blowr*xblow+moxs*fouts-moxp*foutp-
     .             roxlb+moxinj)
      endif
      mnip=intgrl(mniip,wn2b*blowr*xblow+mnis*fouts-mnip*foutp
     .             -rnilb+mniinj)
      mco2p=intgrl(mco2ip,wcb*blowr*xblow+mco2s*fouts-mco2p*foutp
     .             -rcolb)
      map=intgrl(maip,wab*blowr*xblow+mas*fouts-map*foutp)
c  *******       with steam present  ***************
      if(flagst)then
      if(mwvz.gt.(mwl/delt))mwvz=mwl/delt
      mwv=intgrl(mwvzer,mwvz+wwab*blowr*xblow-rwalb+xinj*minjr+
     .            mwv2*fouts-mwv*foutp)
      mwl=intgrl(mwlzer,mwlz)
      uv=intgrl(uvzer,uvz+xinj*minjr*hinj-rwalb*hwv-
     .          (mwv*hwv+htcpgp*1.4*tgp)*foutp+
     .          (mwv2*hwv2+htcpgs*1.4*tgs)*fouts)
      ul=intgrl(ulzer,ulz)
      if(mwl.lt.1.0e-06 .and. ipass.ne.0)then
       ul=0.0
       xic(inoin)=0.0
      endif
      endif
c  ***************************
      if (ico2i .ne. 1)mliop=intgrl(mlioip,-mliop*foutp+(1.+rcmbo)/rcmbo
     .            *cmbro*asli*fra+mlios*fouts-mliop*stick)
      mlc3p=intgrl(mlc3ip,-mlc3p*foutp+(1.+rcmbcs)/rcmbcs*cmbrco*
     .      asli*fra+mlc3s*fouts-mlc3p*stick)
      mlinp=intgrl(mlinip,-mlinp*foutp+(1.+rcmbn)/rcmbn*cmbrn*asli*fra+
     .      mlins*fouts-mlinp*stick)
      mlihp=intgrl(0.,-mlihp*foutp+cmbrw*asli*((1.+rcmbw)/rcmbw-
     .      1./rcmbh2)+mlihs*fouts-mlihp*stick)
      mh2p=intgrl(0.,mh2s*fouts-mh2p*foutp+(1.+rcmbh2)/rcmbh2*
     .      cmbrw*asli)
      outint=intgrl(leako,leak)
      if (flagpn)then
      tpan=intgrl(tpanzo,zz2)
      tins1=intgrl(tins1i,zz8)
      tins2=intgrl(tins2i,zz9)
      endif
      if (flag2)then
      moxs=intgrl(moxis,moxp*foutp-moxs*foutt)
      mnis=intgrl(mniis,mnip*foutp-mnis*foutt)
      mco2s=intgrl(mco2is,mco2p*foutp-mco2is*foutt)
      mas=intgrl(mais,map*foutp-mas*foutt)
      mlios=intgrl(mliois,mliop*foutp-mlios*foutt)
      mlins=intgrl(mlinis,mlinp*foutp-mlins*foutt)
      mlc3s=intgrl(mlc3is,mlc3p*foutp-mlc3s*foutt)
      mlihs=intgrl(0.,mlihp*foutp-mlihs*foutt)
      mh2s=intgrl(0.,mh2p*foutp-mh2s*foutt)
      if(.not.flagst)tgs=intgrl(tgszer,zz3)
      tss=intgrl(tsszer,zzs)
      tfs=intgrl(tfszer,zzfs)
      tehcs=intgrl(tehczs,zzes)
      endif
c *** secondary cell steam effects ********************************
      if(flagst.and.flag2)then
      if(mwvz2.gt.(mwl2/delt))mwvz2=mwl2/delt
      mwv2=intgrl(mwvzr2,mwvz2+xinj2*minjr2+mwv*foutp-mwv2*foutt)
      mwl2=intgrl(mwlzr2,mwlz2)
      uv2=intgrl(uvzer2,uvz2+xinj2*minjr2*hinj2-
     .          (mwv2*hwv2+htcpgs*1.4*tgs)*foutt+
     .          (htcpgp*1.4*tgp+mwv*hwv)*foutp)
      ul2=intgrl(ulzer2,ulz2)
      if(mwl2.lt.1.0e-06 .and. ipass.ne.0)then
       ul2=0.0
       xic(inoin)=0.0
      endif
      endif
c **************************************
      if (flagw)then
      do 1300 i=1,nl
      tc(i)=intgrl(tcic(i),dtcdt(i))
 1300 continue
      endif
      if (flagf)then
      do 1310 i=1,nl1
      tb(i)=intgrl(tbic(i),dtbdt(i))
 1310 continue
      endif
      if (flagc)then
      tcon=intgrl(tsfpi,zzc)
      dcocz=intgrl(0.01,zzd)
      h2left=intgrl(xmh2oi,-relese)
      endif
      if (flagdf) mlead=intgrl(0.,dmpbdt)
      if (flagdf) tlead=intgrl(tleadi,zzpb)
c
      call dynami(time,200)
c
c*********************************************************
c*   post integration section                            *
c    check overp and tli for stop condtion
c    check and correct for lithium and oxygen supply
c*********************************************************
c
  950 continue
      if (thpb .gt. .333*zli) go to 987
      if (tli .ge. tvap) go to 978
      lilp=lit-libp+lilox+lilni
      if (lilp .le. 0.) lilp=0.0
      if (.not. flagpb) zli=lilp/rhli/asli
      alpha=akli/(rhli*cpli)
      if ((lilp .lt. 0.1*lit) .and. (alpha*delt .gt. zli*zli .or. lilp
     .   .lt. 1.0) .and. (.not. flagpb)) flagl=.true.
      if (flagl) lil=lit/10.
      if (.not. flagl .and. .not. flagpb) lil=lilp
      if (tgp .lt. 500. .and. overpp .lt. 1. .and. abs(xmdot)
     .      .lt. 0.1) go to 977
c      if (tli .lt. tmelt) go to 976
      if (icmb .eq. 0 .or. moxp .gt. 0.01) go to 951
      oxlb=oxlfs
      icmb=0
      cmbro=0.0
      roxlb=0.0
  951 continue
      if (ilit .eq. 0 .or. (lit-libp) .ge. 0.01) go to 952
      oxlb=lit/rcmbo
      if (ico2i .eq. 1) colb=lit/rcmbco
      if (ico2i .eq. 1) oxlb=0.0
      ilit=0
      lit=libp
      cmbr=0.0
      cmbro=0.0
      cmbrn=0.0
      cmbrco=0.0
      cmbrc2=0.0
      cmbrw=0.0
      roxlb=0.0
      rnilb=0.0
      rwalb=0.0
      rcolb=0.0
      rc2lb=0.0
  952 continue
      if (mnip .ge. 0.0) go to 953
      mnip=0.0
      icnl=0
      cmbrn=0.
      rnilb=0.0
  953 continue
      if (mwv .ge. 0.0) go to 954
      mwv=0.0
      uv=tgp*htcpgp
      cmbrw=0.0
      rwalb=0.0
  954 continue
      if (mco2p .ge. 0.0) goto 955
      mco2p=0.0
      cmbrco=0.0
      rcolb=0.0
      cmbrc2=0.0
      rc2lb=0.0
      if (ico2i .eq. 1 .and. tli .gt. 1180.) cmbro=0.0
      if (ico2i .eq. 1 .and. tli .gt. 1180.) roxlb=0.0
  955 continue
      cmbrh=3600.*(cmbro+cmbrn+cmbrw+cmbrco+cmbrc2)
      if (cmbrh .ge. 0.1 .or. time .le. 10.) go to 956
      icz=0
      cmbro=0.0
      cmbro1=0.0
      cmbro2=0.0
      cmbrn=0.0
      cmbrw=0.0
      cmbrh=0.0
      cmbrco=0.0
      cmbrc2=0.0
      roxlb=0.0
      rnilb=0.0
      rwalb=0.0
      rcolb=0.0
      rc2lb=0.0
  956 continue
c
c*********************************
c*   convert temp. to deg. f     *
c*********************************
c
      tsfpf=tsfp -460.
      tczf=tcz-460.
      tlif=tli-460.
      tgpf=tgp-460.
      tspf=tsp-460.
      tehcpf=tehcp-460.
      if (flagst) then
       tlpf=tlp-460.
       tlpf2=tlpf2-460.
      endif
      if (.not. flag2) go to 960
      tgsf=tgs-460.
      tfsf=tfs-460.
      tssf=tss-460.
      tehcsf=tehcs-460.
  960 continue
      if (.not. flagpn) go to 961
      tpanf=tpan-460.
      tins1f=tins1-460.
      tins2f=tins2-460.
  961 continue
      if (.not. flagw) go to 1001
      do 1001 i=1,20
      tcf(i)=tc(i)-460.
 1001 continue
      if (.not. flagf) go to 1002
      do 1002 i=1,20
      tbf(i) =tb(i)-460.
 1002 continue
      tconf=tcon-460.
      if (flagdf) tleadf=tlead-460.
      if(flagst)tlpf=tlp-460.
c
c***************************
```

```
c*   time step control     *
c***************************
c
      dt1=abs(relerr*tli/zz1)
      if(flagst.and.uvz.ne.0.0)then
       dt2=abs(relerr*uv/uvz)
      elseif(zz4.ne.0.0)then
       dt2=abs(relerr*tgp/zz4)
      else
       dt2=dtmin
      endif
      dt3=abs(relerr*tsp/zz5)
      if (ilit.eq.0 .or. icz.eq.0) go to 965
      dt5=abs(relerr*tcz/zz6*.05)
      zz99=(cmbrh-cmbrhi)/delt
      if (zz99.eq.0.) go to 965
      dt4=abs(relerr*cmbrh/zz99)
      cmbrhi=cmbrh
      if (ipass.eq.1) dt4=1.e06
      go to 966
  965 continue
      dt4=1.0e06
      dt5=1.0e06
  966 continue
      if (flagdf .and. zzpb .lt. 1.0e-15) zzpb=1.0e-15
      if (flagdf) dt6=abs(relerr*tlead/zzpb)
      bilge=amin1(dt1,dt2,dt3,dt4,dt5)
      if (flagdf) bilge=amin1(bilge,dt6)
      bil=(bilge-delt)/delt
c   this condtion is to remove instability due to steep
c    nitrogen reaction curve
      if (tcz .gt. 1900..and.abs(bil).gt.0.1) then
       delt=delt+(bilge-delt)/10.
      else
       delt=bilge
      endif
c
c*****  test conduction limits on time step   ***
c     limiting conduction rate is determined from pool to pan
c     (if using pan option) otherwise from pool to steel liner
c
      if (flagpn) alpha2=((thkpan+zli)/(zli/akli+thkpan/kpan))/
     .   ((rhli*cpli*zli+rhpan*cppan*thkpan)/(thkpan+zli))
      if (flagpn) pyu=0.075*(thkpan+zli)**2/alpha2
      if (.not. flagpn) alpha2=((thfp+zli)/(zli/akli+thfp/kstlfp))/
     .   ((rhli*cpli*zli+rhsfp*cpsfp*thfp)/(thfp+zli))
      if (.not. flagpn) pyu=0.075*(thfp+zli)**2/alpha2
      if (delt .gt. pyu) delt=pyu
c     conduction test for pool layers if using diffusion model
      if (flagdf .and. delt .gt. pyup) delt=pyup
c   testing two cell exchange rate on time step
      if (.not. flag2 .or. abs(xmdot) .lt. 0.0001) go to 959
      if ((abs(pap-pas)) .lt. .01 .and. delt .gt. .04) delt=.04
      delmp=mairp/abs(xmdot)/250.
      delms=mairs/abs(xmdot)/250.
      if (delt .gt. delmp)  delt=delmp
      if (delt .gt. delms) delt=delms
  959 continue
c    aerosol removal time step check
      if (delt*stick .gt. .40) delt=.40/stick
c    steam effects time step checks
      if(flagst)then
       if(uvz.ne.0.)then
```

```
          deluv=abs(uv/uvz)*relerr
          if(delt.gt.deluv)delt=deluv
         endif
         if(mwvz.ne.0.)then
          delmwv=abs(mwv/mwvz)*relerr
          if(delt.gt.delmwv)delt=delmwv
         endif
         if(ulz.ne.0. .and. mwl.gt.1.0e-06)then
          delul=abs(ul/ulz)*relerr
          if(delt.gt.delul)delt=delul
         endif
         if(mwlz.ne.0.)then
          delmwl=abs(mwl/mwlz)*relerr
          if(delt.gt.delmwl)delt=delmwl
         endif
         if(mwvz2.ne.0.)then
          delmv2=abs(mwv2/mwvz2)*relerr
          if(delt.gt.delmv2)delt=delmv2
         endif
         if(uvz2.ne.0.)then
          deluv2=abs(uv2/uvz2)*relerr
          if(delt.gt.deluv2)delt=deluv2
         endif
         if(mwlz2.ne.0.)then
          delml2=abs(mwl2/mwlz2)*relerr
          if(delt.gt.delml2)delt=delml2
         endif
         if(ulz2.ne.0. .and. mwl2.gt.1.0e-06)then
          delul2=abs(ul2/ulz2)*relerr
          if(delt.gt.delul2)delt=delul2
         endif
        endif
c*********** general time step controls ************************
      if (delt .gt. 0.1 .and. time .lt. 3.0) delt=0.1
      if (delt .gt. 0.2 .and. time .lt. 25.0) delt=0.2
      if (delt .gt. 1.0 .and. time .lt. 2000.0) delt=1.0
c*********** user defined time step controls *********************
      if (delt.lt.dtmin) delt=dtmin
      if (delt .gt. delout) delt=delout
c
c***********************
c*   output section    *
c***********************
c
      if (time.lt.timeo) go to 975
      timeo=timeo+output
      if (ipage.lt.40) go to 974
      write (11,830)  (name(i),i=41,100)
      write (20,1112)
1112  format (//,'  These outputs are the reaction rates in gram Li/min-
     .cm2',//,5x,'   Time    Cmbrhh     Cmbrnh     Cmbroh     Cmbrwh
     . Cmrcoh     Cmrc2h',/)
      write (18,1113)
1113  format (//,' These outputs are the weights of reaction products in
     . LB',//,5x,'Time    Lilox      Lilni      Lilca      Lilc2    Lilcar
     .   Mliop     Mlc3p',/)
      if (flag2) write (12,830)  (name(i),i=101,160)
      if (flagpn) write (13,830)  (name(i),i=161,220)
      write (14,830)  (name(i),i=221,280)
      if (flagdf) write (15,830)  (name(i),i=281,340)
c      write (16,829) i10,i11,i12,i13,i14
c      write (17,833) i15,i16,i17,i18,i19
  830 format(' ',2(20a4,/),//,20a4)
  974 continue
      if (ipage.ge.40) ipage=0
      ipage=ipage+1
      if (time .le. 0.) then
        tval = 0.1
      else
        tval = time
      endif
c     ****  changing RR from lb li/sec-ft2 to gram li/min-cm2
      cmbrnh=cmbrn*3600./122.78
      cmbroh=cmbro*3600./122.78
      cmbrwh=cmbrw*3600./122.78
      cmbrhh=cmbr*3600./122.78
      cmrcoh=cmbrco*3600./122.78
      cmrc2h=cmbrc2*3600./122.78
      if (iflagu .ne. 1) go to 1054
c
c***  this step converts output variables to si  ***
c***  added due to some difficult in comfiling from Sub. si  ***
c
      cmbrh=cmbrh*4.8824
      libp=libp/2.2046
      map=map/2.2046
      mnip=mnip/2.2046
      moxp=moxp/2.2046
      mwap=mwap/2.2046
      mco2p=mco2p/2.2046
      pap=pap/1.450e-01
      tczf=tcz/1.8-273.
      tehcpf=tehcp/1.8-273.
      tgpf=tgp/1.8-273.
      tlif=tli/1.8-273.
      tsfpf=tsfp/1.8-273.
      tspf=tsp/1.8-273.
      zli=zli/3.281
      if (flag2) then
       pas=pas/1.450e-01
       tehcsf=tehcs/1.8-273.
       tgsf=tgs/1.8-273.
       tfsf=tfs/1.8-273.
       tssf=tss/1.8-273.
       xmdot=xmdot/2.2046
      endif
      if (flagst) then
       mwl=mwl/2.2046
       mwl2=mwl2/2.2046
       mwv=mwv/2.2046
       mwv2=mwv2/2.2046
       tlpf=tlp/1.8-273.
       tlpf2=tlp2/1.8-273.
      endif
      if (flagpn) tpanf=tpan/1.8-273.
      if (flagpn) tins1f=tins1/1.8-273.
      if (flagpn) tins2f=tins2/1.8-273.
      if (flagc) tconf=tcon/1.8-273.
      if (flagdf) then
       mlead=mlead/2.2046
       thpb=thpb/3.281
       tleadf=tlead/1.8-273.
       xlidot=xlidot*4.8824*3600.
      endif
      if (flagw) then
       do 1500 i=1,20
       tcf(i)=tc(i)/1.8-273.
 1500 continue
      endif
      if (flagf) then
       do 1490 i=1,20
       tbf(i)=tb(i)/1.8-273.
 1490 continue
      endif
 1054 continue
      write (11,826) time,delt,tczf,tlif,tgpf,pap,tspf,tsfpf
c
c *******this step writes the output into files to be plotted***************
c *** these steps create files for a graphics routine to be plotted on
c *** graphs of the variable indicated vs. time. Plots of other variables
c *** may be created as desired by adding new open (or call link) statements
c *** at the beginning of the code as described in the user's guide.
c
      write (21,1070) time,tspf
      write (22,1070) time,pap
      write (23,1070) time,tgpf
      write (24,1070) time,tsfpf
      write (40,1070) time,tczf
      write (99,1070) time,tlif
       if(flagst)then
c       write(41,1070) time,mwv
c       write(42,1070) time,mwl
c       write(43,1070) time,tlpf
       write(44,1070) time,hratio
       endif
 1070 format (2f10.3)
c*******************************************
c      write (20,849) tval,tlif,pap,tgpf
      num=num+1
      if (flag2) write (12,832) time,tgsf,tfsf,pas,xmdot,moxs,mnis,mco2s
      if (flagpn) write (13,3999) time,tlif,tpanf,tins1f,tins2f,pap
      write (14,831) time,mnip,moxp,mco2p,rn2,ro2,libp
      write (20,1120) time,cmbrhh,cmbrnh,cmbroh,cmbrwh,cmrcoh,cmrc2h
1120  format (3x, f9.1,6e11.4)
      write (18,1114) time,lilox,lilni,lilca,lilc2,lilcar,mliop,mlc3p
1114  format (1x,f7.1,7e10.3)
      if (flagdf) write (15,3999) time,xlidot,tleadf,mlead,thpb,zli
c      write (16,834) time,tbf(i10),tbf(i11),tbf(i12),tbf(i13),tbf(i14)
c      write (17,834) time,tcf(i15),tcf(i16),tcf(i17),tcf(i18),tcf(i19)
  826 format(3x,f9.1,f6.2,f10.2,f10.2,4(1x,f7.2))
 3999 format(3x, f9.1, 5e13.4)
  849 format(f9.1,f10.2,f8.2,e13.4)
  829 format(' nodal concrete floor temperatures',//,5x,'time',
     .       6x,'tb(',i1,')',6x,'tb(',i1,')',6x,'tb(',i1,')',
     .       6x,'tb(',i1,')',6x,'tb(',i1,')')
  831 format(3x,f9.1,3e12.4,2f9.5,e12.4)
  832 format(f8.1,3f8.2,4e11.4)
  833 format(' nodal concrete wall temperatures',//,5x,'time',
     .       6x,'tc(',i1,')',6x,'tc(',i1,')',6x,'tc(',i1,')',
     .       6x,'tc(',i1,')',6x,'tc(',i1,')')
  834 format(f9.1,5f11.2)
      if (iflagu .ne. 1) go to 975
c
c****  this step converts output from si to english after  ****
c       output is printed so that program can calculate
c      things in english again. not needed for output temps.
c
      cmbrh=cmbrh/4.8824
      libp=libp*2.2046
      map=map*2.2046
      mnip=mnip*2.2046
      moxp=moxp*2.2046
      mwap=mwap*2.2046
      mco2p=mco2p*2.2046
      pap=pap*1.450e-01
      zli=zli*3.281
      if (flag2) pas=pas*1.450e-01
      if (flag2) xmdot=xmdot*2.2046
      if (flagdf) xlidot=xlidot/3600./4.8824
      if (flagdf) mlead=mlead*2.2046
      if (flagdf) thpb=thpb*3.281
      if (flagst) then
       mwl=mwl*2.2046
       mwl2=mwl2*2.2046
       mwv=mwv*2.2046
       mwv2=mwv2*2.2046
      endif
 975  continue
      if (time.gt.timef) go to 990
c***** return to top of dynamic cycle *****
      go to 200
c
c***********************
c*     error pointers   *
c***********************
c
  976 continue
      write (11,835)
  835 format(' pool temp. has dropped to lithiums melting temp.')
      go to 990
  977 continue
      write (11,836)
  836 format(' cell gas temp. and press. have returned to normal')
      go to 990
  978 continue
      write (11,837)
  837 format(' lithium temp. above boiling point')
 888  format(1x,e12.4,e12.4)
      go to 990
  979 continue
      write (11,838)
  838 format(1x,'no root found for spray fire for temp.s less than ',
     .        '1 million deg. r')
      go to 990
  980 continue
      write (11,839)
  839 format (' suspended pan option cannot be selected concurrent'/
     .          ' concrete combustion option')
      go to 990
  984 continue
      write (11,844)
  844 format (' spray fire and lithium lead combustion are not',
     .          ' compatible')
      go to 990
  985 continue
      write (11,845)
  845 format(' exx is negative--cannot take root')
      write (11,846) tcz,cmbrh,zz6,zz5,rn2
  846 format(' messed up variables',5e10.3)
      go to 990
  986 continue
      write (11,847) stick,beta
 847 format (' aerosol removal fraction is too large'/
    .       'stick = ',f12.4,'    beta = ',f12.4/)
     go to 990
2001 continue
     write (11,2002)
2002 format(1x,'CO2 atmosphere and gas injec. option cannot be used
    ./1x,concurrently'/)
     go to 990
2003 continue
     write (11,2004)
2004 format(1x,'CO2 atm. and lithium lead combustion opttions cannot
    ./1x,used concurrently'/)
     go to 990
2005 continue
     write (11,2006)
2006 format(1x,'CO2 atm. option is not allowed when oxygen is present'
    .       /)
 987 continue
 848 format(' lead layer thickness is greater than zli/3. diffusion'/
    .' model is no longer valid'/)
 990 continue
     write (11,867)
 867 format(' program execution stopped by program')
     write (11,868) dt1,dt2,dt3,dt4,dt5
     write (10,869) num
 869 format(' ','the number of data points is  ',i4)
 868 format(' values', 5e10.3)
     call exit
     end
```

```
c    these 3 subroutines are designed to be used in a main program which
c    simulates a dynamic system expressed as a set of ode's. these ode's
c    may be reexpressed as a set of integrals which must be integrated
c    simultaneously through the domain of interest starting with the appropriate
c    initial conditions.  for example, the function y may be found from the
c    solution of  dy/dt = rate = f(y,t) and y=y0 at t=t0. this may be
c    rewritten y = intgrl(y0, rate), the open integral of rate over t starting
c    at yo. a set of ode's may be treated in a similiar manner.
c        the main program should consist of two main parts, the initialization
c    section and the dynamic section.  the dynamic section is further divided
c    into integration and post-integration sections.
c        the initial section should be used for input, calculation of necessary
c    constants, and for calculating and setting of initial conditions. it
c    should contain the real intgrl, common, and call init statements.
c        the integration section should start with a numbered continue
c    statement and end with the call dynami  statement.  it should contain
c    all calculations of program variables and non-constant rates. all intgrl
c    function statements should appear in a group immediately preceding the
c    call dynami statement.
c        the integration section will be looped several times during each
c    integration step (simpson's rule uses 4 loops per step, runge-kutta uses
c    5 loops per step).  dynami controls the integration by telling the
c    intgrl function what step it should perform next. the integration
c    variable time is also controled by dynami.  it may or may not be increment-
c    ed during each loop. time should be initialized in the intial section.
c    dynami utilizes multiple returns to control program flow.  the statement
c    number passed to dynami should be that of the first statement in the
```

```
c    integration section. this causes the proper integration looping. at the
c    end of each integration step a normal return is executed and control
c    returns to the first statement following call dynami. this should be
c    the first statement of the post-integration section.
c        because variable values may differ from their true value during the
c    integration looping, all program logic and variable time step calculations
c    executed once at the end of each integration step. time and all variables
c    contained within the integration section will be updated to their 'true'
c    values before control is transfered to the post-integration section.
c    this section should contain at least one if statement which stops program
c    execution. and the last statement should be a go to st.no. where st.no.
c    is the statement number of the first statement in the integration section.
c        approximately 100 integrations may be performed simultaneously.
c
c    variable list
c
c      a   matrix which stores the intermiate values calculated during each loop
c      delt   integration time step
c      dxdt   rate being integrated. calculated using  integral value as
c             returned by intgrl during the previous loop and time set by
c             dynami.  used by intgrl as called for by icount.
c      icount   tells intgrl which integration loop is presently being done
c      imeth = 1 use runge-kutta method
c           = 3 use simpson's rule
c      inoin   tell dynami how many intgrl statements there are in the main
c              program.
c      ipass   tells intgrl to do two special functions during the first two
c              executions of the integration section.
c      istore   tells intgrl where to store the result of its intermediate
c              calculation in matrix a.
c      xic   matrix which store  intial conditions and then is updated to the
c            present integral value at the end of each integration step.
c      xxic   initial condition
c
      subroutine dynami(time,*)
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .               xic(101),a(501)
      if (ipass.eq.0) go to 40
      if (imeth.eq.1) go to 10
c
c    simpson's rule (default) imeth>2
c
      if (icount.eq.4) go to 4
      if (icount.eq.3) go to 3
      time=time+delt/2.
      icount=icount+1
      return 1
    4 continue
      istore=0
      icount=1
      ipass=ipass+1
      inoin=0
      return
    3 continue
      icount =4
      return 1
c
c    runge-kutta method -fixed step-  imeth=1
c
   10 continue
      if (icount.eq.5) go to 4
      if (icount.eq.4) go to 14
      if (icount.eq.2) go to 12
      time=time+delt/2.
      icount=icount+1
      return 1
   12 continue
      icount=3
      return 1
   14 continue
      icount= 5
      return 1
   40 continue
      ipass=1
      return
      end
c    this subroutine initializes variables used by the integration routines.
c    it should be placed in the initialization section of the main program
c    before the first statement of the dynamic section.  see dynami for variable
c    list and integration description.
c
      subroutine init
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .               xic(101),a(501)
      ipass=0
      istore=0
      icount=1
      inoin=0
      return
      end
c
c
c      function intgrl performs the actual integrations. in the main
c    program, all intgrl statements should be placed in a group at the end
c    of the integration section.  all rate calculations should precede this
c    group and it should be immediately followed by the call dynami statement.
c    for variable list and descriptions see dynami.
      real function intgrl(xxic,dxdt)
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .               xic(101),a(501)
      if (ipass.eq.0) go to 40
      istore=istore+1
      if (imeth.eq.1) go to 10
c
c    simpson's rule (default) imeth greater than 2
c
      if (icount.eq.4) go to 4
      if (icount.eq.3) go to 3
      if (icount.eq.2) go to 2
    1 continue
      inoin=inoin+1
      if (ipass.eq.1) xic(inoin)=xxic
      a(istore)=dxdt
      intgrl=xic(inoin)+delt*dxdt/2.
      if(intgrl.lt.0.0)intgrl=0.0
      a(500-istore)=intgrl
      return
    2 continue
      a(istore)=dxdt
      intgrl=a(500+inoin-istore)+delt*dxdt/2.
      if(intgrl.lt.0.0)intgrl=0.0
      return
    3 continue
      intgrl=xic(istore-2*inoin)+delt/6.*(a(istore-2*inoin)+4.*
     .              a(istore-inoin)+dxdt)
      if(intgrl.lt.0.0)intgrl=0.0
      xic(istore-2*inoin)=intgrl
      return
    4 continue
      intgrl=xic(istore-3*inoin)
      if(intgrl.lt.0.0)intgrl=0.0
      return
c
c     runge-kutta method -fixed step-   imeth=1
c
   10 continue
      if (icount.eq.5) go to 15
      if (icount.eq.4) go to 14
      if (icount.eq.3) go to 13
      if (icount.eq.2) go to 12
   11 continue
      inoin=inoin+1
      if (ipass.eq.1) xic(inoin)=xxic
      a(istore)=delt*dxdt
      intgrl=xic(inoin)+.5*a(istore)
      if(intgrl.lt.0.0)intgrl=0.0
      return
   12 continue
      a(istore)=delt*dxdt
      intgrl=xic(istore-inoin)+.5*a(istore)
      if(intgrl.lt.0.0)intgrl=0.0
      return
   13 continue
      a(istore)=delt*dxdt
      intgrl=xic(istore-2*inoin)+a(istore)
      if(intgrl.lt.0.0)intgrl=0.0
      return
   14 continue
      aa=delt*dxdt
      intgrl=xic(istore-3*inoin)+1./6.*(a(istore-3*inoin)+2.*
     .     a(istore-2*inoin)+2.*a(istore-inoin)+aa)
      if(intgrl.lt.0.0)intgrl=0.0
      xic(istore-3*inoin)=intgrl
      return
   15 continue
      intgrl=xic(istore-4*inoin)
      if(intgrl.lt.0.0)intgrl=0.0
      return
   40 continue
      intgrl=xxic
      return
      end
c
c
c     This function is for calculating the RR of Li-Nitrogen
c     reaction which is strongly depending on the lithium pool
c     temp.   This gives the experimental RR of W. Ijams.  This
c     function is used in order to control the RR of Li-N2, since
c     the former version of LITFIRE assumes a continious increase
c     in the RR without limit as the gas arrival rate to the
c     the combustion zone increases.
      function rr(tli)
      tli=tli/1.8-273.
      if (tli .ge. 200. .and. tli .lt. 400.) rr=.03336/200.*(400.-tli)
      if (tli .ge. 400. .and. tli .lt. 450.) rr=
     . .03336+.00416*(450.-tli)/50.
      if (tli .ge. 450. .and. tli .lt. 500.) rr=
     . .03752+.02079*(500.-tli)/50.
      if (tli .ge. 500. .and. tli .lt. 550.) rr=
     . .05831+.02502*(550.-tli)/50.
       if (tli .ge. 550. .and. tli .lt. 600.) rr=
     . .08333+.04167*(600.-tli)/50.
       if (tli .ge. 600. .and. tli .lt. 650.) rr=
     . .125+.08333*(650.-tli)/50.
       if (tli .ge. 650. .and. tli .lt. 700.) rr=
     . .2083+.1917*(700.-tli)/50.
       if (tli .ge. 700. .and. tli .lt. 725.) rr=.4+.225*(725.-tli)/25.
       if (tli .ge. 725. .and. tli .lt. 750.) rr=.625+.125*(750.-tli)/25.
       if (tli .ge. 750. .and. tli .lt. 800.) rr=.75+.0667*(800.-tli)/50.
       if (tli .ge. 800. .and. tli .lt. 950.) rr=.89*sin(tli/600-.175)
       if (tli .ge. 950. .and. tli .lt. 1050.) rr=
     . .875+.015*(950.-tli)/50.
       if (tli .ge. 1050. .and. tli .lt. 1100.) rr=
     . .86+.235*(1050.-tli)/50.
       if (tli .ge. 1100. .and. tli .lt. 1127.) rr=.625*(1127-tli)/27.
       if (tli .gt. 1127. .or. tli .lt. 200.) rr=0.0
       rr=.034106*rr
       tli=(tli+273.)*1.8
       return
       end
c
c
c
       function akexx(t01,t02,rhobar)
       ginbar=32.2
       tbar=0.5*(t01+t02)
       bbar=1.0/tbar
       dbar=((4.94e-05*tbar+0.0188)/(rhobar*3600.))**2
       akbar=(0.014+1.92e-05*(tbar-460.))/3600.
       exbar=(ginbar*bbar*abs(t01-t02)/dbar)**0.3333
       akexx=akbar*exbar
       return
       end
c
c
c    this function is for calculating the partial pressure of lithium in
c         lithium-lead as a function of concentration
       function actvty(xali)
       aliln=8.835*(xali**2.219)-6.0
       if (aliln .gt. 0.0) aliln=0.0
       actvty=xali*exp(aliln)
       return
       end
c
c these subroutines are used to modularize litfire. they include the
c options of two cell geometry and pan geometry as well as floor and
c concrete combustion.
c
c  this is the secondary cell subroutine.
       subroutine cell2
       implicit real (i,k,l,m)
       logical flagn,flagm,flagw,flagf,flag2,flagst
       common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .           flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
       common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i1
       common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .                xic(101),zzz(501)
       common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .               rhli,spill,tli,tlii,zli
       common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .                kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
       common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .               ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .               rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .               tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .               tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .               rair
      common /conop/ c8,cpcon,dtbdt(20),dtcdt(20),gap,kcon,kgap,
     .                l(20),l1(20),nl,nl1,qradb,radb,rhcon,
     .                sflcr,tb(20),tbf(20),tbic(20),tcf(20),
     .                tcic(20),thfc,thwc,tsfpi,tspzer,xsfl,qlflr,qvflr
      common /injop/ dp1,dp2,dp3,mniinj,moxinj,time,vp
      common /secop/ aehcs,c11,c20,chs,cpehcs,cph2,cplih,cpwa,crack,
     .                foutp,fouts,foutt,hinfgs,hinfsg,hingss,hinps,kleak,
     .                leak,mairp,mairs,mais,mas,mh2s,mlihs,mlinis,mlins,
     .                mliois,mlios,mniis,mnis,moxis,moxs,mwais,
     .                mlc3s,mlc2s,mcs,mco2s,rholc3,rholc2,rhocar,
     .                mwas,pap,pas,paszer,ra,rbreak,rholih,
     .                rholin,rholio,rwpgas,tehcs,tehcsf,tehczs,tgsf,
     .           tfsf,tgszer,tssf,vs,xmdot,xmehcs,xmola,zz3,zzfs
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .                ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .                phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +                mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .                xmolp,cell
      common /steam2/cpa2,cpvb2,emgs,hfg2,hlp2,htcpgs,hum2,mwl2,
     .                mwlv2,mwlz2,mwlzr2,mwv2,mwvv2,mwvz2,mwvzr2,
     .                phase2,ph2o2,ph2ob2,rair2,tlp2,ul2,ulp2,ulz2,
     .                ulzer2,uv2,uvz2,uvzer2,uwv2,vg2,vl2,vlp2,vvb2,
     .                vvg2,xg2,xmols,stmin2,stout2,xinj2,rhoas
      common /heat2/ hinecs
c
      if (flagn) n=1
      go to (1,2,3,4,5)n
    1 continue
c
c**********     read in secondary cell parameters and      ********
c                       initial conditions
c
      read (3,701) vs,chs,paszer,tgszer,tsszer,tfszer
      read (3,701) crack,hum2,wo2s,was,cpas,wco2s
      read (3,701) tehczs,xmehcs,aehcs,cpehcs,hinecs
      read (3,701) estlws,cpsws,kstlws,rhsws,aws,thws
      read (3,701) estlfs,cpsfs,kstlfs,rhsfs,afs,thfs
      if (iswich .eq. 1) read (3,701) tswich
c
      write (10,800) chs,vs,wo2s,hum2,was,cpas,crack,wco2s
      write (10,801) tehczs,xmehcs,aehcs,cpehcs,hinecs
      write (10,802) tgszer,tsszer,tfszer,paszer
      write (10,803) estlws,cpsws,kstlws,rhsws,aws,thws
      write (10,804) estlfs,cpsfs,kstlfs,rhsfs,afs,thfs
      if (iswich .eq. 1) write (10,810) tswich
c
  700 format(20a4)
  701 format (6f12.4)
  800 format (' secondary containment dimensions'/1x,32(1h-)//t10,
     .'chs =   ',f12.4,t35,'vs =    ',f12.4,t60,'wo2s =   ',f12.4//t10,
     .'hum2 =  ',f12.4,t35,'was =   ',f12.4,t60,'cpas =   ',f12.4//t10,
     .'crack = ',f12.4,t35,'wco2s = ',f12.4//)
  801 format(' extraneous heat capacity node data'/1x,33(1h-)//t10,
     .'tehczs =',f12.4,t35,'xmehcs =',f12.4,t60,'aehcs = ',f12.4//
     .t10,'cpehcs =',f12.4,t35,'hinecs =',f12.4//)
  802 format (' secondary initial conditions'/1x,28(1h-)//t10,
     .'tgszer =',f12.4,t35,'tsszer =',f12.4,t60,'tfszer =',f12.4//t10,
     .'paszer =',f12.4//)
  803 format (' secondary steel wall data'/1x,25(1h-)//t10,
     .'estlws =',f12.4,t35,'cpsws = ',f12.4,t60,'kstlws =',f12.4//t10,
     .'rhsws = ',f12.4,t35,'aws =   ',f12.4,t60,'thws =  ',f12.4//)
  804 format (' secondary steel floor data'/1x,26(1h-)//t10,
     .'estlfs =',f12.4,t35,'cpsfs = ',f12.4,t60,'kstlfs =',f12.4//t10,
     .'rhsfs = ',f12.4,t35,'afs =   ',f12.4,t60,'thfs =  ',f12.4//)
  810 format (' closing of crack between primary and secondary cells
     . is'/'allowed when time is greater than tswich =',f11.2//)
c
      n=2
      return
    2 continue
c*********  initialize secondary cell containment variables   ********
      data breaks,foutp,fouts,foutt,mh2s,mlihs,mlinis,mlins,mliois,mlios
     .     ,mlc3s,mlc3is,rbreak,xmdot,zz3,zzes,ul2,ulz2,uv2,uvz2,
     .      mwl2,mwlz2,mwv2,mwvz2/24*0.0/
c
      flagm=.false.
      gamma=1.4
      cd=1.
      tehcs=tehczs
      tss=tsszer
      thfs=thfp
      tfs=tfszer
      tgs=tgszer
c
c *** initialize the amount of water vapor in the secondary cell ***
      flagn=.true.
      if(flagst) call steam2
      flagn=.false.
c *******************************************************************
c
      wn2s=1.-wo2s-was-wco2s
      xmols=1./(wo2s/32.+wn2s/28.+was/xmola+wco2s/44.)
      rins=1545./xmols
      rhoais=paszer*144./rins/tgszer
      if(flagst)rhoais=(paszer-ph2o2)*144./rins/tgszer
      rhoas=rhoais
      mniis=wn2s*rhoais*vs
      mnis=mniis
      moxis=wo2s*rhoais*vs
      moxs=moxis
      mais=was*rhoais*vs
      mas=mais
      mco2is=wco2s*rhoais*vs
      mco2s=mco2is
      if(flagst)wwas=mwv2/(mnis+moxis+mas+mwv2)
c
c**** determine the inital energy of the secondary vapor region ***
      if(flagst)call steam2
c
c**********     conversion to ft. - lb. - sec.     **********
c
      crack=crack/144.
      kstlws=kstlws/3600.
      kstlfs=kstlfs/3600.
      n=3
      return
    3 continue
c *** zero temperature rates of change **********************
      zzes=0.0
      zzfs=0.0
      zzs=0.0
```

```
*****     compute physical properties dependent on temperature  .*****
c*****  calculate air composition and specific heat at const. volume   ******
      mairs=moxs+mnis+mh2s+mas+mco2s
      xmols=(28.*mnis+32.*moxs+2.*xmh2s+xmola*mas+44.*mco2s)/mairs
      rair2=1545./xmols
      rhoas=mairs/vs
      foxs=moxs/mairs
      fwas=mwv2/(mwv2+mairs)
      fnis=mnis/mairs
      fco2s=mco2s/mairs
      cpo2s=(0.184+3.2e-06*tgs-1.36e04/(tgs*tgs))
      cpmoxs=cpo2s*moxs
      cpn2s=(0.172+8.57e-06*tgs+1.02e-09*tgs*tgs)
      cpco2s=.201
      cpmcos=cpco2s*mco2s
      cplc3s=.761
      cpmlcs=cplc3s*mlc3s
      cpmnis=cpn2s*mnis
      cplios=0.0602*tgs**.326
      cplins=0.3368+3.67e-04*tgs
      cpmlos=cplios*mlios
      htcpgs=cpmoxs+cpmnis+cpmlos+cpas*mas+cplins*mlins+cplih*mlihs+
     .   cph2*mh2s+cpmcos+cpmlcs
      cpa2=htcpgs/(mairs+mlins+mlihs+mlc3is)
c***** calculating radiative interchange factors   *****
      emgs=1.-exp(-(mlios/rholio+mlins/rholin+mlihs/rholih+mlc3s/rholc3)
     .     *2.27e05*chs/vs/ra)
      if (emgs .le. 0.005) emgs=0.005
c
c *** calculate emissivity and temperature of secondary cell gas ***
      if(flagst)call steam2
c **************************************************************
c
      rifps=1./((1.-estlwp)/estlwp+(1.-estlws)/estlws*(awp/aws)+
     .   (1.+awp/aws)/(1.+awp/aws*(1.-emgs)))
      rifpga=(estlwp*emgs)/((1.-estlwp)*emgs+estlwp)
      riffps=1./((1.-estlfp)/estlfp+(1.-estlfs)/estlfs*(asli/afs)+
     .   (1.+asli/afs)/(1.+asli/afs*(1.-emgs)))
      riffgs=(estlfp*emgs)/((1.-estlfp)*emgs+estlfp)
      rifscw=(estlws*emconc)/(estlws+emconc-estlws*emconc)
      rifscf=(estlfs*emconc)/(estlfs+emconc-estlfs*emconc)
c
c *** check for boiling and condensation, calculate heat transfer ******
      if(flagst)call steam2
c
c *** calculating gas heat transfer coefficients  (without steam) ******
      if(.not.flagst .or. mwv.le.0.0)then
c  secondary gas to secondary extraneous heat capacity
      hehcs=hinecs*akexx(tgs,tehcs,rhoas)
c  secondary steel liner to secondary gas
      hsec=hingss*akexx(tgs,tss,rhoas)
c  primary steel wall liner to secondary containment gas
      hwpgas=hinps*akexx(tsp,tgs,rhoas)
c  primary steel floor liner to secondary containment gas
      hfpgas=hinfgs*akexx(tsfp,tgs,rhoas)
c  secondary steel floor to secondary cell gas
      hfsgas=hinfsg*akexx(tfs,tgs,rhoas)
      endif
c  secondary steel liner to ambient (superceded by concrete to ambient
c      if concrete option in use)
      if (.not. flagw) ha=hinsam*akexx(tss,ta,.074)
c   secondary steel floor liner to ambient
      if (.not. flagf) hamf=hinfam*akexx(tfs,ta,.074)
```

```
  100 continue
c ****   calculate gas heat transfer coefficients with steam present ***
      if(flagst)call steam2
c
c*****   calculating thermal diffusivities between nodes   *****
      c11=kstlws*ha/(rhsws*cpsws*thws*(kstlws+thws*ha/2.))
      c12=kstlfs*hamf/(rhsfs*cpsfs*afs*(kstlfs+thfs*hamf/2.))
      c14=kstlfs*hfsgas/(rhsfs*cpsfs*thfs*(thfs*hfsgas/2.+kstlfs))
      c15=kstlfs*hfsgas*afs/htcpgs/(thfs*hfsgas/2.+kstlfs)
      c18=kstlfp*hfpgas/(rhsfp*cpsfp*thfp*(thfp*hfpgas/2.+kstlfp))
      c19=kstlfp*hfpgas*asli/htcpgs/(thfp*hfpgas/2.+kstlfp)
      c20=kstlwp*hwpgas/(rhswp*cpswp*thwp*(thwp*hwpgas/2.+kstlwp))
      c21=kstlws*hsec/(rhsws*cpsws*thws*(thws*hsec/2.+kstlws))
      c22=kstlwp*hwpgas*awp/htcpgs/(thwp*hwpgas/2.+kstlwp)
      c23=kstlws*hsec*aws/htcpgs/(thws*hsec/2.+kstlws)
      cehcgs=hehcs*aehcs/htcpgs
      cgsehc=hehcs*aehcs/xmehcs/cpehcs
c
c *** steam injection into secondary cell option *****************
      xinj2=0.0
      if(flagst.and.(time.ge.stmin2 .and. time.le.stout2))xinj2=1.
c
c*****   calculating radiative heat transfer between nodes   *****
      qradps=sigma*awp*(tsp**4-tss**4)*rifps
      rwpws=qradps/(thwp*awp*rhswp*cpswp)
      rwswp=qradps/(thws*aws*rhsws*cpsws)
      qradfs=sigma*asli*(tsfp**4-tfs**4)*riffps
      rfpfs=qradfs/(thfp*asli*rhsfp*cpsfp)
      rfsfp=qradfs/(thfs*afs*rhsfs*cpsfs)
      qradpg=sigma*awp*(tsp**4-tgs**4)*rifpga
      rwpgas=qradpg/(thwp*awp*rhswp*cpswp)
      rspgs=qradpg/htcpgs
      qradfg=sigma*asli*(tsfp**4-tgs**4)*riffgs
      rfpgas=qradfg/(thfp*asli*rhsfp*cpsfp)
      rgasfp=qradfg/htcpgs
      n=4
      return
    4 continue
c***** calculating radiation from outer steel liners   *****
      if (.not. flagw) qradc=sigma*aws*(tss**4-ta**4)*estlws
      if (flagw) qradc=sigma*aws*(tss**4-tc(1)**4)*rifscw
      radc=qradc/(thws*aws*rhsws*cpsws)
      if (.not. flagf) qradb=sigma*afs*(tfs**4-ta**4)*estlfs
      if (flagf) qradb=sigma*afs*(tfs**4-tb(1)**4)*rifscf
      radb=qradb/(thfs*afs*rhsfs*cpsfs)
c* modifying primary steel wall and floor temperature rates of change
      zz5=zz5-c20*(tsp-tgs)-rwpws-rwpgas
      if(mwv2.gt.0.0)zz5=zz5-c20*(tsp-tgs)
      zz7=zz7+csbli*(tli-tsfp)-c18*(tsfp-tgs)-rfpfs-rfpgas
      if(mwv2.gt.0.0)zz7=zz7-c18*(tsfp-tgs)
      if((mwl2*vlp2/afs).gt.0.1)zz7=zz7+rfpfs
calculate extraneous heat capacity temperature rate of change
      zzes=zzes+cgsehc*(tgs-tehcs)
      if(mwv2.gt.0.0)zzes=zzes-cgsehc*(tgs-tehcs)
calculate outer cell gas temperature rate of change   deg r/sec
      if(.not.flagst)then
       zz3=breaks+rspgs+c22*(tsp-tgs)+c23*(tss-tgs)
     .    +cehcgs*(tehcs-tgs)+c19*(tsfp-tgs)+rgasfp+c15*(tfs-tgs)
      else
       uvz2=uvz2+qradpg+qradfg
       if(mwv.le.0.0)uvz2=uvz2+c22*(tsp-tgs)*htcpgs+c23*(tss-tgs)*htcpgs
     .      +cehcgs*(tehcs-tgs)*htcpgs+c19*(tsfp-tgs)*htcpgs
     .      +rgasfp*htcpgs+c15*(tfs-tgs)*htcpgs
```

```
      endif
calculate outer wall steel temperature rate of change   deg r/sec
      if (.not. flagw) zzs=zzs+c21*(tgs-tss)-c11*(tss-ta)+rwswp-radc
      if (flagw) zzs=zzs+c21*(tgs-tss)-c7*(tss-tc(1))+rwswp-radc
      if (mwv2.gt.0.0) zzs=zzs-c21*(tgs-tss)
calculate outer floor steel temperature rate of change  deg r/sec
      if (.not. flagw) zzfs=zzfs+c14*(tgs-tfs)-c12*(tfs-ta)+rfsfp-radb
      if (flagw) zzfs=zzfs+c14*(tgs-tfs)-c8*(tfs-tb(1))+rfsfp-radb
      if (mwv2.gt.0.0) zzfs=zzfs-c14*(tgs-tfs)
      if ((mwl2*vlp2/afs).gt.0.1) zzfs=zzfs-rfsfp
      n=5
      return
    5 continue
c***********************************
c**    calculating overpressure   **
c***********************************
      xmairs=moxs/32.+mnis/28.+mas/xmola+mco2s/44.
      pas=1545.*xmairs*tgs/144./vs
      if(flagst)pas=pas+ph2o2
      overps=pas-paszer
c***********************************
c**    calcu. total  leakage    ***
c***********************************
      leak=kleak*(abs(pas-14.7))**0.5
      if (pas .lt. 14.7) leak=0.
      if (abs(pap-pas) .lt. 0.0006 .and. iswich .eq. 1 .and.
     .    time .gt. tswich) crack=0.0
      if (crack .eq. 0.0 .and. iswich .eq. 1) write (11,815) time
  815 format (' cell pressures have equilized at time = ',f11.2/
     .  'crack size has been set to zero for remainder of calculation')
      if (crack .eq. 0.0) iswich=0
      if (crack .eq. 0.0) go to 112
      if (abs(pap-pas) .lt. 0.0006) go to 106
      if (pap-pas) 101,106,107
c*****  flow out of secondary into primary  *****c
  101 foutp=0.
      if (pap/pas .ge. 0.53) go to 103
c*****  sonic  *****c
      if (flagm) go to 102
c*****  first time sonic  *****c
      write (12,816)
      ipage=ipage+1
      flagm=.true.
  102 xmdot=cd*crack*12.*sqrt(0.94*gin*pas*rhoas)
      if(flagst)xmdot=cd*crack*12.*sqrt(0.94*gin*pas*(rhoas+1./vg2))
      go to 105
c*****  subsonic  *****c
  103 if (.not. flagm) go to 104
c*****  first time back to normal subsonic  *****c
      write (12,817)
      ipage=ipage+1
      flagm=.false.
  104 xmdot=cd*crack*sqrt(2.*gin*(pas-pap)*rhoas)*12.
      if(flagst)xmdot=cd*crack*sqrt(2.*gin*(pas-pap)*(rhoas+1./vg2))*12.
  105 fouts=xmdot/mairs
      if(flagst)fouts=xmdot/(mairs+mwv2)
      rbreak=xmdot*(gamma*tgs-tgp)/(mairp+mwv+delt*xmdot)
      breaks=xmdot*tgs*(1.-gamma)/(mairs+mwv2-delt*xmdot)
      go to 112
c*****   no flow   *****c
  106 foutp=0.
      fouts=0.
      xmdot=0.
      rbreak=0.
      breaks=0.
      go to 112
c*****  flow out of primary into secondary  *****c
  107 fouts=0.
      if (pas/pap .ge. 0.53) go to 109
c*****   sonic   *****c
      if (flagm) go to 108
c*****  first time sonic   *****c
      write (12,816)
      ipage=ipage+1
      flagm=.true.
  108 xmdot=cd*crack*12.*sqrt(0.94*gin*pap*rhoap)
      if(flagst)xmdot=cd*crack*12.*sqrt(0.94*gin*pap*(rhoap+1./vg))
      go to 111
c*****   subsonic   *****c
  109 if (.not. flagm) go to 110
c*****  first time back to normal subsonic   *****c
      write (12,817)
      ipage=ipage+1
      flagm=.false.
  110 xmdot=cd*crack*sqrt(2.*gin*(pap-pas)*rhoap)*12.
      if(flagst)xmdot=cd*crack*sqrt(2.*gin*(pap-pas)*(rhoap+1./vg))*12.
  111 foutp=abs(xmdot)/(mairp+mwv)
      rbreak=abs(xmdot)*tgp*(1.-gamma)/(mairp+mwv-delt*abs(xmdot))
      breaks=abs(xmdot)*(gamma*tgp-tgs)/(mairs+mwv2+delt*abs(xmdot)),
      xmdot=0.-xmdot
  816 format (' flow between primary and secondary has become sonic')
  817 format(' flow between primary and secondary has returned to subson
     .ic')
  112 continue
      foutt=fouts+leak
c *** calculate hydrogen buildup in secondary cell ***
      hrat2=mh2s/2./(xmairs+mwv2/18.)
c
      n=3
      return
      end
c
c
c
c  this is the pan geometry subroutine.
      subroutine pan
      implicit real (i,k,l,m)
      integer tavhi,tavlo,tfhi,tflo
      logical flagn,flag2,flagst
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .         flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spili,tli,tlii,zli
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .               kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
      common /panop/ ains,apan,bredth,clist,cpins,cppan,emgp,fpg,fpw,
     .               kpan,rhins,rhpan,thkin1,thkin2,thkpan,
     .               tins1,tins1f,tins1i,tins2,tins2f,tins2i,
     .               tpan,tpanf,tpanzo,zz2,zz4,zz8,zz9
```

```
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .               ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .               phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +               mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .               xmolp,cell
      common /stmpn/ tsat,hsat,huch
c
      if (flagn) n=1
      go to (1,2,3)n
    1 continue
c
c***********     read in pan geometry parameters     ************
c                (only if using pan option)
c
      read (4,701) kpan,rhpan,cppan,rhins,cpins,emins
      read (4,701) tpanzo,apan,bredth,ains,hingpf
      read (4,701) thkpan,thkin1,thkin2
c
      write (10,800) tpanzo,apan,cppan,thkpan,bredth,kpan,rhpan
      write (10,801) thkin1,thkin2,ains,rhins,cpins,emins,hingpf
c
  700 format(20a4)
  701 format (6f12.4)
  800 format(///,' data for suspended pan optional geometry:',/,1x,
     .41(1h-),//t10,'tpanzo =',f12.4,t35,'apan =  ',f12.4,t60,
     .'cppan = ',f12.4//t10,'thkpan =',f12.4,t35,'bredth =',f12.4//t10,
     .'kpan =  ',f12.4,t35,'rhpan = ',f12.4//)
  801 format(//t10,'thkin1 =',f12.4,t35,'thkin2 =',f12.4,t60,'ains =  '
     ..,f12.4//t10,'rhins = ',f12.4,t35,'cpins = ',f12.4,t60,
     .'emins = ',f12.4//t10,'hingpf =',f12.4//)
c
      n=2
      return
    2 continue
c**********  initialize pan geometry variables    *********
      fpg=0.23
      fpw=0.384
      tins1i=0.5*(tpanzo+tgpzer)
      tins2i=tgpzer
      tins1=tins1i
      tins2=tins2i
c  convert thermal conductivity of li pan to btu/sec-ft-deg R
      kpan=kpan/3600.
      n=3
      return
    3 continue
c
c*****  compute physical properties dependent on temperature   *****
c
c*****      radiative interchange factors    *****
      rifpas=1./((1.-emins)/emins+(1.-estlfp)/estlfp*ains/afp+
     .   (ains/afp+1.)/(1.+ains/afp*(1.-emgp)))
      rifpag=emins*emgp/(emins+emgp-emins*emgp)
c
c***********     calculating gas heat transfer coefficients    **********
      hfpgp=hingpf*akexx(tgp,tsfp,rhoap)
      hpan=0.714*hb
c
c  *** modifications to heat transfer due to steam condensation **
c
      if(flagst.and. mwv.gt.0.0)then
c*** liquid pool to primary floor *****************
      if(mwl.gt.1.0e-06)then
```

```
c determine average fluid properties
       tave=(tlp+tfp)/2.
       tavhi=int(tave/20.)-23.
       tavlo=tavhi-1.
       intdsr=(tave/20.)-int(tave/20.)
       mulv=(25.3/((tave/1.8)**2.+91.*tave/1.8-8.58e04))/1.488
       cplv=(sat(tavhi,9)-sat(tavlo,9))*intdsr+sat(tavlo,9)
       vlpv=(sat(tavhi,3)-sat(tavlo,3))*intdsr+sat(tavlo,3)
       nulv=mulv*vlpv
       kwat=(0.686-5.87e-06*(abs(tave/1.8-415.))**2.+7.3e-10*pap*6895.)
       kwat=kwat/1.73/3600.
       if(tave.gt.1165.)then
        nulv=1.46e-06
        kwat=8.667e-05
        cplv=1.368
        vlpv=sat(34,3)
       endif
       tfhi=int(tfp/20.)-23.
       tflo=tfhi-1.
       intdsf=(tfp/20.)-int(tfp/20.)
       vlpf=(sat(tfhi,3)-sat(tflo,3))*intdsf+sat(tflo,3)
       if(tfp.gt.1165.)vlpf=sat(34,3)
       betaf=abs(1./vlp*(vlpf-vlp)/(tfp-tlp2))
c determine h from GrPr
       grprf=32.2*betaf*abs(tfp-tlp)*afp**1.5
     . /vlpv/kwat*cplv/nulv
       if((tfp.gt.tlp).and.((1.e05.le.grprf).and.
     .   (grprf.le.2.e07)))then
        ca=0.54
        aa=0.25
       elseif((tfp.gt.tlp).and.(2.0e07.lt.grprf).and.
     .        (grprf.le.3.0e10))then
        ca=0.14
        aa=0.333
       elseif(grprf.gt.3.0e10)then
        ca=0.021
        aa=0.4
       elseif((tfp.lt.tlp).and.(3.0e05.le.grprf).and.(grprf.le.3.e10))
     .  then
        ca=0.27
        aa=0.25
       elseif(tlp.eq.tfp)then
        ca=0.0
        aa=1.0
       else
        ca=1.0
        aa=1.0
        grprf=1.0
       endif
       hlpflr=kwat/afp**.5*ca*grprf**aa
       zh2o=mwl*vlp/afp
       if(hlpflr.lt.(2.*kwat/zh2o))hlpflr=2.*kwat/zh2o
       qlflr=2.*(tlp-tfp)/(thfp/kstlfp+1./hlpflr)
       ulz=ulz-qlflr
       zz7=zz7+qlflr/rhsfp/afp/thfp/cpsfp
c
      else
c
c     primary gas to primary floor--no liquid water
       qvflr=2.*(tsat-tsfp)*afp/(thfp/kstlfp+2./huch)
       if(tsfp.gt.tsat)then
        rhogp=rhoap+1./vg
        hfpgp=hingpf*akexx(tgp,tsfp,rhogp)
```

```
         qvflr=2.*(tgp-tsfp)*afp/(thfp/kstlfp+2./hfpgp)
        endif
        mcondf=qvflr/hfg
         if(tsfp.gt.tsat.or.qvflr.le.0.0)mcondf=0.0
        mwvz=mwvz-mcondf
        mwlz=mwlz+mcondf
        uvz=uvz-qvflr
        ulz=ulz+mcondf*hsat
        zz7=zz7+qvflr/rhsfp/afp/thfp/cpsfp
       endif
c gas to pan insulation
       qvpan=ains/(thkin2/2./kin2+1./huch)*(tsat-tins2)
       if(tins2.gt.tsat)then
        hpan=0.714*hb
        qvpan=ains/(thkin2/2./kin2+1./hpan)*(tgp-tins2)
       endif
       mcondp=qvpan/hfg
       if(tins2.gt.tsat .or. qvpan.le.0.0)mcondp=0.
       mwvz=mwvz-mcondp
       mwlz=mwlz+mcondp
       uvz=uvz-qvpan
       ulz=ulz+mcondp*hsat
       zz9=zz9+qvpan/thkin2/rhins/cpins/ains
       endif
c
c*****  calculations with suspended lithium spill pan   *****
c
       aht=asli+zli*bredth
       tet1=0.0025*(tins1-460.)-2.5
       kin1=(.70892+.36584*tet1+.04565*tet1**2-.00791*tet1**3)/43200.
       tet2=0.0025*(tins2-460.)-2.5
       kin2=(.70892+.36584*tet2+.04565*tet2**2-.00791*tet2**3)/43200.
       if(.not.flagst)ypagas=ains/(thkin2/2./kin2+1./hpan)
       c2=ypagas/htcpgp
       c13=ypagas/(rhins*ains*thkin2*cpins)
       c16=kstlfp*hfpgp/(rhsfp*cpsfp*thfp*(thfp*hfpgp/2.+kstlfp))
       c17=kstlfp*hfpgp*afp/htcpgp/(thfp*hfpgp/2.+kstlfp)
       qrads=sigma*ains*(tins2**4-tsfp**4)*rifpas
       qradcg=sigma*ains*(tins2**4-tgp**4)*rifpag
       rpanst=qrads/(rhsfp*afp*thfp*cpsfp)
       rstpan=qrads/(rhins*ains*thkin2*cpins)
       rgaspa=qradcg/(rhins*ains*thkin2*cpins)
       rpagas=qradcg/htcpgp
       clipan=2.*aht/(lil*cpli)/(zli/akli+thkpan/kpan)
       cpanli=2.*aht/(rhpan*apan*thkpan*cppan)/(zli/akli+thkpan/kpan)
       cpnin1=2./(rhpan*apan*thkpan*cppan)/(thkpan/kpan/apan+thkin1/
     .    kin1/ains)
       cin1pn=2./(rhins*ains*thkin1*cpins)/(thkpan/kpan/apan+thkin1/
     .    kin1/ains)
       cin12=2./(rhins*cpins*thkin1)/(thkin1/kin1+thkin2/kin2)
       cin21=cin12*thkin1/thkin2
c****modifying primary cell temperature rates of change due to pan  ***
       zz1=zz1+clist*(tli-tsfp)-clipan*(tli-tpan)
       if(.not.flagst)zz4=zz4+c2*(tins2-tgp)+rpagas+c17*(tsfp-tgp)
       if(flagst)uvz=uvz+qradcg
       zz7=zz7-csbli*(tli-tsfp)+c16*(tgp-tsfp)+rpanst
       if(flagst)zz7=zz7-c16*(tgp-tsfp)
       if(flagst.and.(mwl*vlp/afp).gt.0.1)zz7=zz7-rpanst
c
c *****   calculate li spill pan temp. rate of change  deg r/sec  *****
       zz2=cpanli*(tli-tpan)+cpnin1*(tins1-tpan)
c         calculate insulation temperature rate of change
       zz8=cin1pn*(tpan-tins1)+cin12*(tins2-tins1)
```

```
      zz9=zz9+cin21*(tins1-tins2)+c13*(tgp-tins2)-rstpan-rgaspa
      if(flagst)zz9=zz9-c13*(tgp-tins2)
      if(flagst.and.(mwl*vlp/afp).gt.0.1)zz9=zz9+rstpan
      return
      end
c  this is the wall concrete subroutine
      subroutine concw
      implicit real (i,k,l,m)
      integer i,iam
      dimension  c4(20)
      logical flagn,flag2
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spili,tli,tlii,zli
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .               kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .               xic(101),zzz(501)
      common /conop/ c8,cpcon,dtbdt(20),dtcdt(20),gap,kcon,kgap,
     .               l(20),l1(20),nl,nl1,qradb,radb,rhcon,
     .               sflcr,tb(20),tbf(20),tbic(20),tcf(20),
     .               tcic(20),thfc,thwc,tsfpi,tspzer,xsfl,qlflr,qvflr
c
      if (flagn) n=1
      go to (1,2,3)n
    1 continue
      i10=1
      i11=int(.25*nl)
      i12=int(.5*nl)
      i13=int(.75*nl)
      i14=nl
      nlm1=nl-1
c*****   initialize wall concrete variables   *****
      data c3,c5,c7,radcc/4*0.0/
      if (flag2) go to 100
      aws=awp
      cpsws=cpswp
      kstlws=kstlwp
      rhsws=rhswp
      thws=thwp
      tsszer=tspzer
  100 continue
      do 1001 iam=1,20
      c4(iam)=0.
 1001 dtcdt(iam)=0.
      do 1002 i=1,nl
      tcic(i)=tsszer
      tc(i)=tsszer
 1002 l(i)=thwc*l(i)
      n=2
      return
    2 continue
c***** calculating gas heat transfer coefficient from outermost *****
c                concrete node to ambient
      tcnl=tc(nl)
      ha=hinsam*akexx(tcnl,ta,.074)
c   *****     calculating thermal diffusivities between nodes     *****
      usuba=kcon*ha/(kcon+ha*l(nl)/2.)
      b=l(1)/(kcon*2.)+gap/kgap+thws/(kstlws*2.)
      c3=1./(b*l(1)*rhcon*cpcon)
      do 1004 i=1,nlm1
      c4(i)=2.*kcon/(rhcon*cpcon*l(i)*(l(i)+l(i+1)))
 1004 continue
      c5=usuba/(rhcon*cpcon*l(nl))
      c7=1./(b*thws*rhsws*cpsws)
      n=3
      return
    3 continue
      if (.not. flag2) tss=tsp
      radcc=qradc/(l(1)*aws*rhcon*cpcon)
c*****         wall concrete teperature change         ***********
      dtcdt(1)=c3*(tss-tc(1))+c4(1)*(tc(2)-tc(1))+radcc
      dtcdt(nl)=c4(nlm1)*(tc(nlm1)-tc(nl))-c5*(tc(nl)-ta)
      do 1006 i=2,nlm1
 1006 dtcdt(i)=c4(i)*(tc(i+1)-tc(i))+c4(i-1)*(tc(i-1)-tc(i))
      n=2
      return
      end
c
c
c  this is the floor concrete subroutine
      subroutine concf
      implicit real (i,k,l,m)
      integer i,iam,ib
      dimension  c10(20)
      logical flagn,flag2,flagdf,flagst
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spill,tli,tlii,zli
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .               kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .               xic(101),zzz(501)
      common /conop/ c8,cpcon,dtbdt(20),dtcdt(20),gap,kcon,kgap,
     .               l(20),l1(20),nl,nl1,qradb,radb,rhcon,
     .               sflcr,tb(20),tbf(20),tbic(20),tcf(20),
     .               tcic(20),thfc,thwc,tsfpi,tspzer,xsfl,qlflr,qvflr
c
      if (flagn) n=1
      go to (1,2,3)n
    1 continue
      i15=1
      i16=int(.25*nl)
      i17=int(.5*nl)
      i18=int(.75*nl)
      i19=nl
      if (flag2) go to 100
      afs=asli
      cpsfs=cpsfp
      kstlfs=kstlfp
      rhsfs=rhsfp
      thfs=thfp
      tfszer=tsfpi
  100 continue
      nl1m1=nl1-1
c********   initialize floor concrete variables   ********
      data c8,c9,radcb/3*0.0/
c
      do 1001 iam=1,20
      c10(iam)=0.
 1001 dtbdt(iam)=0.
      do 1003 i=1,nl1
      tbic(i)=tfszer
      tb(i)=tfszer
 1003 l1(i)=thfc*l1(i)
      n=2
      return
c
    2 continue
c   *****     calculating thermal diffusivities between nodes     *****
      bb=l1(1)/(kcon*2.)+gap/kgap+thfs/(kstlfs*2.)
      c8=1./(bb*thfs*rhsfs*cpsfs)
      c9=1./(bb*l1(1)*rhcon*cpcon)
      do 1005 i=1,nl1m1
      c10(i)=2.*kcon/(rhcon*cpcon*l1(i)*(l1(i)+l1(i+1)))
 1005 continue
      n=3
      return
    3 continue
      if (.not. flag2) tfs=tsfp
      radcb=qradb/(l1(1)*afs*rhcon*cpcon)
c*****          floor concrete temperature change
      dtbdt(1)=c9*(tfs-tb(1))+c10(1)*(tb(2)-tb(1))+radcb
      dtbdt(nl1)=c10(nl1m1)*(tb(nl1m1)-tb(nl1))
      do 1007 ib=2,nl1m1
 1007 dtbdt(ib)=c10(ib)*(tb(ib+1)-tb(ib))+c10(ib-1)*(tb(ib-1)-tb(ib))
      n=2
      return
      end
c
c
cthis is the gas injection subroutine
      subroutine injec
      implicit real (i,k,l,m)
      logical flagn,flagas
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /injop/ dp1,dp2,dp3,mniinj,moxinj,time,vp
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
c
      if (flagn) n=1
      go to (1,2)n
    1 continue
c
c*****   read in gas injection variables  *****
c      (only if using gas injection option)
      read (5.700) tone,ttwo,tthree,dp1,dp2,dp3,fct1,fct2,fct3
  700 format (3f10.2,6f8.4)
c
      write (10,800) tone,ttwo,tthree,dp1,dp2,dp3,fct1,fct2,fct3
  800 format (///' data for gas injection modeling:',/,1x,31(1h-),
     .//t10,'tone =  ',f12.4,t35,'ttwo =  ',f12.4,t60,'tthree =',f12.4
     .//t10,'dp1 =   ',f12.4,t35,'dp2 =   ',f12.4,t60,'dp3 =   ',f12.4
     .//t10,'fct1 =  ',f12.4,t35,'fct2 =  ',f12.4,t60,'fct3 =  ',f12.4)
c
      injec1=0
      injec2=0
      injec3=0
      n=2
      return
    2 continue
c*****   injection of nitrogen and oxygen to model hedl experiment  ***
      if (time .lt. tone .or. time .gt. (tone+60.)) go to 100
      if (injec1 .eq. 0 .and. dp1 .gt. 0.0) write (11,801) tone,dp1
  801 format (/,' injection of gas at time = ',f8.0,' to raise
     .      pressure by',f8.4,'  psi.')
      injec1=1
      moinj1=2.9822*vp/tgp*dp1*(1.0-fct1)
      mninj1=2.6094*vp/tgp*dp1*fct1
      moxinj=moinj1/60.
      mniinj=mninj1/60.
  100 continue
      if (time .lt. ttwo .or. time .gt. (ttwo+60.)) go to 101
      if (injec2 .eq. 0 .and. dp2 .gt. 0.0) write (11,801) ttwo,dp2
      injec2=1
      moinj2=2.9822*vp/tgp*dp2*(1.0-fct2)
      mninj2=2.6094*vp/tgp*dp2*fct2
      moxinj=moinj2/60.
      mniinj=mninj2/60.
  101 continue
      if (time .lt. tthree .or. time .gt. (tthree+60.)) go to 102
      if (injec3 .eq. 0 .and. dp3 .gt. 0.0) write (11,801) tthree,dp3
      injec3=1
      moinj3=2.9822*vp/tgp*dp3*(1.0-fct3)
      mninj3=2.6094*vp/tgp*dp3*fct3
      moxinj=moinj3/60.
      mniinj=mninj3/60.
  102 continue
      if (time .gt. (tthree+60.)) flagas=.false.
      return
      end
c
c
c
cthis is the concrete combustion subroutine
      subroutine concc
      implicit real (i,k,l,m)
      logical flagn,flagd
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spill,tli,tlii,zli
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
      common /conop/ c8,cpcon,dtbdt(20),dtcdt(20),gap,kcon,kgap,
     .              l(20),l1(20),nl,nl1,qradb,radb,rhcon,
     .              sflcr,tb(20),tbf(20),tbic(20),tcf(20),
     .              tcic(20),thfc,thwc,tsfpi,tspzer,xsfl,qlflr,qvflr
      common /ccop/ cmbro,cracon,dcocz,h2left,qcconc,rcmbo,rcmbw,
     .              relese,tcigni,tcon,tconf,xmh2oi,zzc,zzd,zzdin
c
      if (flagn) n=1
      go to (1,2,3)n
    1 continue
c***** read in concrete combustion parameters *******
      read (4,700) zzdin,qcconc,cracon,xmh2oi,tcigni,rcmbc
  700 format (6f12.4)
c
      write (10,800) zzdin,qcconc,cracon,xmh2oi,tcigni,rcmbc
  800 format (//' concrete combustion input data'/1x,30(1h-)//t10,
     .'zzdin = ',f12.4,t35,'qcconc =',f12.4,t60,'cracon =',f12.4//
     .t10,'xmh2oi =',f12.4,t35,'tcigni =',f12.4,t60,'rcmbc = ',f12.4//)
c
      n=2
      return
    2 continue
      data ccocoz,ccoczp,ccozco,cpcocz,relese,zzc/6*0.0/
      zzd=zzdin
      tcon=tsfpi
      dcocz=0.01
      xmcocz=1.0
      flagd=.false.
      h2left=xmh2oi
      vconc=afp*l1(1)
      n=3
      return
    3 continue
c  water release from concrete  --- correlation based on drying tests
c  of magnitite.  see r.d. peak "caceco  a containment analysis code-
c  users guide"
      relese=0.
      if (tb(1) .ge. 658.5 .and. tb(1) .lt. 1960.)  water=(1.-exp(26.207
     .  +tb(1)*(-0.0721+tb(1)*(6.96e-05-tb(1)*2.26e-08)))/11.7)*xmh2oi
c  "water" is the amount that should be left at tb(1) in units of lbs./ft**3
      if (tb(1) .ge. 658.5 .and. (h2left-water) .gt. 0. .and. tb(1)
     .    .lt. 1960.) relese=(h2left-water)*vconc/30.
      if (tb(1) .ge. 1960. .and. h2left .gt. 0.) relese=h2left*vconc/30.
c  in other words the release rate of water is such that the difference
c  between the actual amount and the correct amount ( according to the
c  correlation used) is given off in thirty seconds.
c*****      calculate thermal diffusivities     *****
      xmcocz=dcocz*cracon*rhcon
      cpcocz=2.*cracon*kcon*akli/(kcon*zli+akli*dcocz)/xmcocz
      ccoczp=2.*cracon*kcon*akli/(kcon*zli+akli*dcocz)/lil
      ccocoz=2.*cracon*kcon/(dcocz+l1(1))/xmcocz
      ccozco=2.*cracon*kcon/(dcocz+l1(1))/(rhcon*cpcon*l1(1)*afp)
c  flagd is true when concrete combustion stops
      flagd=.false.
      if (lilp .lt. 0.1 .or. tcon .lt. tcigni) flagd=.true.
      zzd=zzdin
      if (flagd) zzd=0.0
      zzc=cpcocz*(tli-tcon)+ccocoz*(tb(1)-tcon)+zzd*cracon*qcconc*rhcon
     .    /xmcocz/cpcon+relese*qcw*rcmbw/xmcocz/cpcon
      zz1=zz1+ccoczp*(tcon-tli)
      dtbdt(1)=dtbdt(1)+ccoczo*(tcon-tb(1))
      cmbro=relese*rcmbo+zzd*cracon*rhcon*rcmbc
      n=3
      return
      end
c
c
c     this is the lithium lead combustion subroutine
c
c
      subroutine lipb
      implicit real (i,k,l,m)
      logical flagn,flagl,flagdf,flagst
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spill,tli,tlii,zli
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lead/ cplead,klead,rhlead,mlipb,xalloy,atml,atmpb,cmbr
      common /pbpool/ dmpbdt,zzpb,mlead,tleadi,xwli,dflipb,xlidot,
     .                thpb,tleadf,foo
      common /pbdif/ cczp,cgli,clig,cpcz,cpmcz,dfilm,kfilm,pyup,
     .               qradp,rczp,rgli,rifczp,rifpg,rifpw,rlig,rwli,
     .               tlead,yapcz,zz6,dflvar
c
      if (flagn) n=1
      go to (1,2,3)n
    1 continue

      foo = 1.0
c
c*****    read in lead parameters     *****
c
      read (4,701) cplead,klead,rhlead,alloyi,qdiss,cplipb
      read (4,702) dflvar
c
      write (10,800) cplead,klead,rhlead,alloyi,qdiss,cplipb,dflvar
c
  701 format (6f12.4)
  702 format (e12.5)
  800 format(//,' data for lithium lead combustion option:',/,1x,40(1h-)
     .,//t10,'cplead =',f12.4,t35,'klead =',f12.4,t60,'rhlead =',f12.4//
     .t10,'alloyi =',f12.4,t35,'qdiss =',f12.4,t60,'cplipb =',f12.4//
     .t10,'dflvar =',e12.5//)
c
      klead=klead/3600.
      atmlpb=spill/(6.941*alloyi+(1.-alloyi)*207.2)
      atmpb=(1.-alloyi)*atmlpb
      atmli=alloyi*atmlpb
      mlipbi=spill
      spill=atmlpb*6.941*alloyi
c
      write (10,801) spill
  801 format(' modified parameters for lithium in lithium lead pool',/,
     .1x,52(1h-),//t10,' amount of lithium available for combustion =',
     .f12.4//t10,' thickness of lipb pool is less than zli above and',/
     .t35,' is calculated in program')
      n=2
      return
c
    2 continue
c*****    modifying lithium pool properties to include lead     *****
c
      if (flagdf) go to 100
      mlipb=mlipbi-libp
      xmlipb=mlipbi-lit+lilp
      atml=atmli-libp/6.941
      if (atml .le. 0.0) atml=0.0
      xalloy=atml/(atml+atmpb)
      go to 110
  100 continue
      mlipb=mlipbi-libp-mlead
      if (mlipb .lt. 0.0) mlipb=0.0
      xmlipb=mlipb
      atml=atmli*mlipb/mlipbi
      xalloy=alloyi
  110 continue
      xwli=xalloy*6.941/(xalloy*6.941+(1.-xalloy)*207.2)
      akli=xwli*akli+(1.-xwli)*klead-0.72*abs(akli-klead)*xwli*(1.-xwli)
      cpli = cplipb
      if (xalloy .eq. 0.17) then
        rhli = 648.65
      else
        rhli=xalloy*rhli+(1.-xalloy)*rhlead+332.6*xalloy*(1.-xalloy)**0.64
      endif
      zli=xmlipb/rhli/asli
      if ((mlipb .lt. 0.1*mlipbi) .and. (alpha*delt .gt. zli*zli .or.
     .   xmlipb .lt. 1.0)) flagl=.true.
      if (flagl) lil=mlipbi/10.
      if (.not. flagl) lil=xmlipb
      n=3
      if (foo .eq. 1.0) then
        write (10,11191) akli,cpli,rhli,zli
11191   format (' akli     cpli  .     rhli      zli     '//4F12.6)
        foo = 0.0
      endif
      return
c
    3 continue
c*****     modifying pool temp rate of change to include heat of lithium     *****
c                dissociation from lead
      zz1=zz1-qdiss*cmbr*asli/(lil*cpli)
      n=2
      return
      end
c
c
c    this is the lithium lead diffusion model subroutine.
c
      subroutine lidiff
      implicit real (i,k,l,m)
      logical flagn,flagpb,flagisi,flagdf,flagst
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spill,tli,tlii,zli
      common /lead/ cplead,klead,rhlead,mlipb,xalloy,atml,atmpb,cmbr
      common /injop/ dp1,dp2,dp3,mniinj,moxinj,time,vp
      common /pbpool/ dmpbdt,zzpb,mlead,tleadi,xwli,dflipb,xlidot,
     .                thpb,tleadf,foo
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .               kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
      common /panop/ ains,apan,bredth,clist,cpins,cppan,emgp,fpg,fpw,
     .                  kpan,rhins,rhpan,thkin1,thkin2,thkpan,
     .                  tins1,tins1f,tins1i,tins2,tins2f,tins2i,
     .                  tpan,tpanf,tpanzo,zz2,zz4,zz8,zz9
      common /units/ aehcp,beta,chp,cmbrh,cpap,cpehcp,map,mnip,
     .                  moxp,mwap,papzer,qcn,qco,qco1,qco2,qcw,qvap,
     .                  tcz,tczf,tczi,tehcp,tehcpf,tehczp,tgpf,
     .                  tlif,tmelt,tsfpf,tspf,tvap,xmehcp
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .                  xic(101),zzz(501)
      common /pbdif/ cczp,cgli,clig,cpcz,cpmcz,dfilm,kfilm,pyup,
     .                  qradp,rczp,rgli,rifczp,rifpg,rifpw,rlig,rwli,
     .                  tlead,yapcz,zz6,dflvar
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .                  ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .                  phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +                  mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .                  xmolp,cell
c
      if (flagn) n=1
      go to (1,2,3)n
    1 continue
      tleadi=tlii
      zzpb=0.
      n=2
      return
c
    2 continue
      thpb=mlead/rhlead/asli
      if (thpb .lt. 1.0e-16) thpb=1.0e-16
      dflipb=dflvar*exp(-1224./tli)!this is the diffusion coefficient (6.5E-08)
      xlidot=dflipb*rhli*xwli/thpb
      dmpbdt=(1.-xalloy)/xalloy*cmbr*asli*207.2/6.941
      n=3
      return
    3 continue
c
      zli1=.667*zli
      zli2=.333*zli
      klipb1=(mlead*klead+.333*lil*akli)/(mlead+.223*lil)
      cplpb1=(mlead*cplead+.333*lil*cpli)
      thpb1=zli2+thpb
c
c*****      modify pool, combustion zone and primary cell temp rates of change
  100 continue
      if (icz .eq. 0) go to 110
      zz1=zz1-cczp*(tcz-tli)-rczp+qvap*cmbr*asli*cczp/yapcz+rwli+rgli
      if(flagst)uvz=uvz-qradz
      if(.not.flagst)zz4=zz4-rlig
      zz5=zz5-rliw
      zz6=zz6+cpcz*(tcz-tli)+qradp/cpmcz
c
      cclipb=2.*asli*klipb1*akli/
     .        (.667*lil*cpli*(zli1*klipb1+thpb1*akli))
      ccpbli=2.*klipb1*akli*asli/(cplpb1*(zli1*klipb1+thpb1*akli))
      yapcz=kfilm*klipb1*asli/(dfilm*klipb1+kfilm*thpb1/2.)
      cpcz=yapcz/cpmcz
      cczp=yapcz/cplpb1
      qradp=sigma*asli*(tcz**4-tlead**4)*rifczp
      rczp=qradp/cplpb1
      qrady=sigma*asli*(tlead**4-tsp**4)*rifpw
      qradz=sigma*asli*(tlead**4-tgp**4)*rifpg
      rliw=qrady/(thwp*awp*rhswp*cpswp)
      rwli=qrady/cplpb1
      rgli=qradz/cplpb1
      rlig=qradz/htcpgp
c
      zzpb=cczp*(tcz-tlead)+rczp-qvap*cmbr*asli/cplpb1
     .     -rwli-rgli-ccpbli*(tlead-tli)
      zz1=zz1+cclipb*(tlead-tli)
      if(flagst)uvz=uvz+qradz
      if(.not.flagst)zz4=zz4+rlig
      zz5=zz5+rliw
      zz6=zz6-qradp/cpmcz-cpcz*(tcz-tlead)
      go to 120
  110 continue
c***** modify temps without combustion zone modeling *****
      zz1=zz1-cgli*(tgp-tli)+rwli+rgli
      if(flagst)uvz=uvz-yalig*(tli-tgp)-qradz
      if(.not.flagst)zz4=zz4-clig*(tli-tgp)-rlig
      zz5=zz5-rliw
c
      yalig=klipb1*hb*asli/(klipb1+hb*thpb1/2.)
      clig=yalig/htcpgp
      qradw=sigma*asli*(tlead**4-tsp**4)*rifpw
      qradg=sigma*asli*(tlead**4-tgp**4)*rifpg
      rliw=qradw/(thwp*awp*rhswp*cpswp)
      rwli=qradw/cplpb1
      rgli=qradg/cplpb1
      rlig=qradg/htcpgp
      cgli=yalig/cplpb1
      cclipb=2.*asli*klipb1*akli/
     .        (.667*lil*cpli*(zli1*klipb1+thpb1*akli))
      ccpbli=2.*klipb1*asli*akli/(cplpb1*(zli1*klipb1+thpb1*akli))
c
      zzpb=cgli*(tgp-tlead)-rwli-rgli-ccpbli*(tlead-tli)
      zz1=zz1+cclipb*(tlead-tli)
      if(flagst)uvz=uvz+yalig*(tlead-tgp)+qradg
      if(.not.flagst)zz4=zz4+clig*(tlead-tgp)+rlig
      zz5=zz5+rliw
      zz6=(tli-tcz)/delt
  120 continue
      alphap=((thpb1+zli1)/(zli1/akli+thpb1/klipb1))/
     .   (((rhli*cpli*zli1)+(cplpb1/asli))/(thpb1+zli1))
      pyup=0.075*(thpb1+zli1)**2/alphap
      n=2
      return
      end
c
c this is the subroutine which sets CO2 only atmosphere option.
c
      subroutine co2 (icmb,icni,ico2i,n)
      icmb=0
      icni=0
      ico2i=1
      if (n .eq. 1) write (10,100)
  100 format(' CO2 atmosphere option is selected'/,1x,33(1h_)/)
      return
      end
c
c this is the subroutine for the steam in containment option
c
      subroutine steam
      implicit real (i,k,l,m)
      integer p,tlo,thi
      logical flagn,lts(2),fr
      dimension ts(2)
      real nulv
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .               ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .               phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +               mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .               xmolp,cell
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spill,tli,tlii,zli
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .               kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /units/ aehcp,beta,chp,cmbrh,cpap,cpehcp,map,mnip,mco2p,
     .               moxp,mwap,papzer,qcn,qco,qco1,qco2,qcw,qvap,qcc,
     .               ql2c2,tcz,tczf,tczi,tehcp,tehcpf,tehczp,tgpf,
     .               tlif,tmelt,tsfpf,tspf,tvap,xmehcp
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
      common /injop/ dp1,dp2,dp3,mniinj,moxinj,time,vp
      common /secop/ aehcs,c11,c20,chs,cpehcs,cph2,cplih,cpwa,crack,
     .               foutp,fouts,foutt,hinfgs,hinfsg,hingss,hinps,kleak,
     .               leak,mairp,mairs,mais,mas,mh2s,mlihs,mlinis,mlins,
     .               mliois,mlios,mniis,mnis,moxis,moxs,mwais,
     .               mlc3s,mlc2s,mcs,mco2s,rholc3,rholc2,rhocar,
     .               mwas,pap,pas,paszer,ra,rbreak,rholih,
     .               rholin,rholio,rwpgas,tehcs,tehcsf,tehczs,tgsf,
     .          tfsf,tgszer,tssf,vs,xmdot,xmehcs,xmola,zz3,zzfs
      common /conop/ c8,cpcon,dtbdt(20),dtcdt(20),gap,kcon,kgap,
     .               l(20),l1(20),nl,nl1,qradb,radb,rhcon,
     .               sflcr,tb(20),tbf(20),tbic(20),tcf(20),
     .               tcic(20),thfc,thwc,tsfpi,tspzer,xsfl,qlflr,qvflr
      common /panop/ ains,apan,bredth,clist,cpins,cppan,emgp,fpg,fpw,
     .               kpan,rhins,rhpan,thkin1,thkin2,thkpan,
     .               tins1,tins1f,tins1i,tins2,tins2f,tins2i,
     .               tpan,tpanf,tpanzo,zz2,zz4,zz8,zz9
      common /stmpn/ tsat,hsat,huch
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .               xic(101),zzz(501)
      common /heat/  hingsp,hinecp
c
      if (flagn)ns=1
      go to (1,2,3,4,5)ns
    1 continue
c
c************** initialize the amount of water vapor in containment ***
c
      tlo=int(tgpzer/20.)-24.
      thi=tlo+1.
      intdis= (tgpzer/20.)-int(tgpzer/20.)
      psat= (sat(thi,2)-sat(tlo,2))*intdis+sat(tlo,2)
      ph2o= psat*hum
      vvg= (sat(thi,4)-sat(tlo,4))*intdis+sat(tlo,4)
      mwv= vp*hum/vvg
      mwvzer=mwv
      mwvv= mwv
      mwlzer=mwl
      vg=1.0e20
      if(hum.gt.0.0)vg= vvg/hum
      uwv= (sat(thi,6)-sat(tlo,6))*intdis+sat(tlo,6)
      ns=2
      return
    2 continue
c
c **************************************************
c
c determine the initial energy of the vapor region
      mairp=mnip+moxp+map+mco2p
       cpn2p=(0.172+8.57e-06*tgp+1.02e-09*tgp*tgp)
       cpo2p=(0.184+3.2e-06*tgp-1.36e04/tgp/tgp)
       cpco2p=0.29
      ua= (mnip*cpn2p+moxp*cpo2p+map*cpap+mco2p*cpco2p)*tgp
      uv= ua+uwv*mwv
      uvzer=uv
      ulzer=ul
      ns=3
      return
    3 continue
c
c************** calculate the emissivity of the air-vapor mixture ***
c
       if((ph2o/14.7*chp/4.).le. 0.5)then
       ew1= 0.37*(ph2o/14.7*chp/4.)**0.67
       else
       ew1= 0.28+0.11*alog(ph2o/14.7*chp/4.)
       endif
       estair= 0.0476*(ph2o+pap)*ew1
       if(estair.gt.(1.4*ew1))estair= 1.4*ew1
       emgp=emgp+estair
       if(emgp.gt.1.0)emgp=1.0
      if (emgp .le. 0.005) emgp=0.005
c
c ********* determining primary air-vapor mixture temperature ***
      htcair=mairp*cpa
      if(mwv.le.0.)go to 70
      lts(1)=.false.
      lts(2)=.false.
      fr=.true.
      ts(1)=491.0
      ts(2)=1940.0
      if(tgp.lt.ts(1))tgp=ts(1)
      vg=(vp-vl)/mwv
 51   tgps=tgp
      cell=1.
      call proptv
 52   temp=uv-htcair*tgp-mwv*uwv
 55   j=2
      if(temp.gt.0.)j=1
      ts(j)=tgp
      lts(j)=.true.
      if(fr) go to 57
      if(abs(temp/uv).le.0.0005)go to 56
      if(.not.(lts(1).and.lts(2)))go to 58
      if(abs(ts(1)-ts(2)).le.0.005)go to 56
 58   temp3= temp2-temp
      if(temp3.eq.0.)go to 1902
      temp4= deltat/temp3
      if(temp4.le.0.)go to 1902
      deltat=temp*temp4
 61   j=j+1
      if(j.eq.3)j=1
      temp3=ts(j)-tgp
      if(abs(deltat).le.abs(temp3))go to 60
      if(lts(j))go to 59
      deltat=temp3
      tgp=ts(j)
      go to 64
 59   deltat=0.9*temp3
 60   tgp=tgp+deltat
 64   temp2=temp
      go to 51
 57   fr=.false.
      deltat=sign(0.5,temp)
      go to 61
 56   if(phase.eq.2.)xg=1.0
      mwvv=xg*mwv
      mwlv=mwv-mwvv
      pap=ph2o+mairp*rair*tgp/(vp-vl)/144.
      go to 1903
 1902 deltat=0.5*(ts(1)+ts(2))-tgp
      go to 60
 70   ph2o=0.0
      mwv=0.0
      mwvv=0.0
      mwlv=0.0
      tgp=uv/htcpgp
      pap=mairp*rair*tgp/vp/144.
 1903 continue
      ns=4
      return
    4 continue
c *********************************************
c******** check for boiling or condensation if water is present *******
      if(mwl.gt.1.0e-06)then
       ulp=ul/mwl
       cell=1.
       call proptu
c check for pool boiling
       do 1480 n=1,34
        p=n+1.
        if(pap.lt.sat(p,2))then
         intdsp=(pap-sat(n,2))/(sat(p,2)-sat(n,2))
         ulpb=(sat(p,5)-sat(n,5))*intdsp+sat(n,5)
         ugpb=(sat(p,6)-sat(n,6))*intdsp+sat(n,6)
         go to 444
        endif
 1480  continue
 444   continue
        boil=0.
       if(ulp.gt.ulpb)then
        boil=1.
        mboil=(ul-mwl*ulpb)/(ugpb-ulpb)
        uvz=uvz+ugpb*mboil/delt
        mwlz=mwlz-mboil/delt
        mwvz=mwvz+mboil/delt
        ulz=ulz-ugpb*mboil/delt
       endif
       vl=mwl*vlp
c
c without boiling, check for evaporation and condensation
c   q"=hb(tgp-tlp)+[kb.18(hfg+hf)(mfvg-mfvb)]/xam
c   is the expression for heat transferred during evaporation
c
      if(boil.eq.0.)then
       mfvg=mwv/(vp-vl)/18./(mwv/(vp-vl)/18.+mairp/(vp-vl)/xmolp)
       if(mfvg.eq.1.0)mfvg=0.9999999
      mfag=1.-mfvg
c at the liquid-vapor boundary:
      mfvb=1./vvb/18./(1./vvb/18.+(pap-ph2ob)*144./rair/tlp/xmolp)
      if(mfvb.eq.1.0)mfvb=0.9999999
      mfab=1.-mfvb
      mflog=mfab/mfag
      xam=(mfab-mfag)/alog(mflog)
c determine h from the product GrPr
c find steam and air properties at the boundary and in the bulk vapor
      do 1470 n=1,2
       temp=abs(tlp/1.8)
       if(n.eq.1)temp=abs(tgp/1.8)
       kair=.02665+6.013e-05*(temp-310.93)
       kair=kair/1.731/3600
       muair=1.912e-05+4.152e-08*(temp-310.93)
       muair=muair/1.488
       vvt=vvb
       if(n.eq.1)vvt=vvg
       kh2o=17.6+5.87e-02*(temp-273.)+1.04e-04*(temp-273.)**2
     . -4.51e-08*(temp-273.)**3+(103.51+.4198*(temp-273.)-2.771e-05*
     . (temp-273.)**2)*16.02/vvt*.001+2.148e08/(temp-273.)**4.2/
     . (vvt/16.02)**2
       kh2o=kh2o*.001/1.729/3600.
       muv=11.4/(temp**2-884*temp+1.36e06)
       muv=muv/1.488
       if(n.eq.2)then
        kairb=kair
        muairb=muair
        kh2ob=kh2o
        muvb=muv
       else
        kairg=kair
        muairg=muair
        kh2og=kh2o
        muvg=muv
       endif
 1470 continue
      temp2=(muairb/muvb)**.5
      phiawb=.218942*(1.+.88808*temp2)**2
      phiwab=.277605*(1.+1.2603/temp2)**2
      vab=1.0e20
      if(pap.gt.ph2ob)vab=1./rhoap
      vsb=vvb*ph2ob/pap
      mub=vsb*muairb/(vsb+vab*phiawb)+vab*muvb/(vab+vsb*phiwab)
      betab=1./tlp
      rhob=1./vab+1./vsb
      cpb=(vsb*cpa+vab*cpvb)/(vab+vsb)
      kbndry=vsb*kairb/(vsb+vab*phiaw)+vab*kh2ob/(vab+vsb*phiwa)
      asurf=(afp-asli)
c  calculate the heat transfer from GrPr
      grpr=betab*rhob*cpb*32.2*abs(tgp-tlp)/kbndry*asurf**1.5/mub
      dab=2.478e-04*(14.22/pap)*(tgp/460.)**1.81
      if(tgp.lt.tlp)then
       if(grpr.lt.2.e07)then
       ca=.54
       aa=.25
       else
       ca=.14
       aa=.3333
       endif
      elseif(tgp.gt.tlp)then
       ca=.27
       aa=.25
       elseif((vvb.eq.(vp-vl)/mwvv).and.(tgp.eq.tlp))then
        go to 1999
       else
        ca=0.
        aa=1.0
       endif
c calculate the sensible heat transfer coefficient
       hsens=ca*kbndry/asurf**.5*grpr**aa
       mgb=18.*mfvb+xmolp*mfab
       prsc=cpb*rhob*dab/kbndry
       kb=hsens/cpb/mgb*(prsc)**.666
c without a temperature difference, the mass transfer coefficient must be
c calculated differently than with one
       if(tgp.eq.tlp) kb=1.02*pap*dab/(asurf**.5*1545.*tlp)*
     .  (asurf**1.5*32.2*abs(mwvv/(vp-vl)-1./vvb)/mub/dab)**.373
c
       a=kb*mgb*cpb*(mfvg-mfvb)/hsens/xam
c mass condensation/evaporation rate
       condr=kb*18.*(mfvg-mfvb)/xam*asurf
       if(condr.lt.(-1.*mwl/delt))condr=-1.*mwl/delt
c account for greater heat transfer with large mass transfer rates
       hprime=(a/(exp(a)-1.))*hsens
c heat transfer rate
       hcond=hprime*(tgp-tlp)+condr/asurf*(hlp+hfg)
c transfer heat and mass to and from liquid and vapor regions
       mwvz=mwvz-condr
       mwlz=mwlz+condr
       uvz=uvz-hcond*asurf
       ulz=ulz+hcond*asurf
      endif
 1999 continue
      endif
c
c**********     calculating gas heat transfer coefficients    **********
c
c determine the effects of steam on the heat transfer coefficients
c find the saturation temperature and the Uchida heat transfer coeff.
      if(mwv.gt.0.0)then
       do 1460 n=1,34
       p=n+1.
       if(ph2o.lt.sat(p,2))then
       intdsp=(ph2o-sat(n,2))/(sat(p,2)-sat(n,2))
       tsat=intdsp*(sat(p,1)-sat(n,1))+sat(n,1) +460.
       hsat=intdsp*(sat(p,7)-sat(n,7))+sat(n,7)
       go to 1111
       endif
 1460  continue
 1111  continue
       mr=mairp/mwv
       do 1450 n=1,14
       p=n+1
       if(mr.gt.hu(p,1))then
       intdsh=(mr-hu(p,1))/(hu(n,1)-hu(p,1))
       huch= hu(p,2)-intdsh*(hu(p,2)-hu(n,2))
       go to 2222
       endif
 1450  continue
 2222  continue
       if(mr.ge.50.)huch=2.
       if(mr.le.0.1)huch=280.
       huch=huch/3600.
      endif
      ns=5
      return
    5 continue
c ***calculate heat transfer coefficients with steam present***
      if(mwv.gt.0.0)then
       rhogp=rhoap+1./vg
c
c  gas to wall
      qu=2.*(tsat-tsp)*awp/(thwp/kstlwp+2./huch)
      if(tsp.gt.tsat)then
       hgwp=hingsp*akexx(tgp,tsp,rhogp)
       qu=2.*(tgp-tsp)*awp/(thwp/kstlwp+2./hgwp)
      endif
      mcondw=qu/hfg
      if(tsp.gt.tsat .or. qu.le.0.0)mcondw=0.
      mwvz=mwvz-mcondw
      mwlz=mwlz+mcondw
      uvz=uvz-qu
      ulz=mcondw*hsat+ulz
      zz5=zz5+qu/rhswp/awp/thwp/cpswp
c
c
c gas to extraneous heat capacity
      qvehc=huch*(tsat-tehcp)*aehcp
      if(tehcp.gt.tsat)then
       hehcp=hinecp*akexx(tgp,tehcp,rhogp)
       qvehc=(tgp-tehcp)*hehcp*aehcp
      endif
      mconde=qvehc/hfg
      if(tehcp.gt.tsat .or. qvehc.le.0.0)mconde=0.
      mwvz=mwvz-mconde
      mwlz=mwlz+mconde
      uvz=uvz-qvehc
      ulz=ulz+mconde*hsat
      zzep=zzep+qvehc/xmehcp/cpehcp
c
      endif
c
      ns=3
      return
      end
c
c ****************************************
c
c this is the subroutine for the steam in the secondary cell option
c
      subroutine steam2
      implicit real (i,k,l,m)
      integer p,tavhi,tavlo,tfhi,tflo,thi,tlo
      logical flagn,lts(2),fr
      dimension ts(2)
      real nulv
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .          flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .               ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .               phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +               mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .               xmolp,cell
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .               kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .                tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .                tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .                rair
      common /secop/ aehcs,c11,c20,chs,cpehcs,cph2,cplih,cpwa,crack,
     .                foutp,fouts,foutt,hinfgs,hinfsg,hingss,hinps,kleak,
     .                leak,mairp,mairs,mais,mas,mh2s,mlihs,mlinis,mlins,
     .                mliois,mlios,mniis,mnis,moxis,moxs,mwais,
     .                mlc3s,mlc2s,mcs,mco2s,rholc3,rholc2,rhocar,
     .                mwas,pap,pas,paszer,ra,rbreak,rholih,
     .                rholin,rholio,rwpgas,tehcs,tehcsf,tehczs,tgsf,
     .            tfsf,tgszer,tssf,vs,xmdot,xmehcs,xmola,zz3,zzfs
      common /steam2/cpa2,cpvb2,emgs,hfg2,hlp2,htcpgs,hum2,mwl2,
     .                mwlv2,mwlz2,mwlzr2,mwv2,mwvv2,mwvz2,mwvzr2,
     .                phase2,ph2o2,ph2ob2,rair2,tlp2,ul2,ulp2,ulz2,
     .                ulzer2,uv2,uvz2,uvzer2,uwv2,vg2,vl2,vlp2,vvb2,
     .                vvg2,xg2,xmols,stmin2,stout2,xinj2,rhoas
      common /intgl/ imeth,icount,istore,inoin,ipass,delt,
     .                xic(101),zzz(501)
      common /heat2/ hinecs
c
      if (flagn)ns=1
      go to (1,2,3,4,5)ns
    1 continue
c
c********** initialize the amount of water vapor in the secondary cell ***
c
      tlo=int(tgszer/20.)-24.
      thi=tlo+1.
      intdis= (tgszer/20.)-int(tgszer/20.)
      psat2= (sat(thi,2)-sat(tlo,2))*intdis+sat(tlo,2)
      ph2o2= psat2*hum2
      vvg2= (sat(thi,4)-sat(tlo,4))*intdis+sat(tlo,4)
      mwv2= vs*hum2/vvg2
      mwvzr2=mwv2
      mwvv2= mwv2
      mwlzr2=mwl2
      vg2=1.0e20
      if(hum2.gt.0.0)vg2= vvg2/hum2
      uwv2= (sat(thi,6)-sat(tlo,6))*intdis+sat(tlo,6)
      ns=2
      return
    2 continue
c
c *** determine the initial energy of the secondary cell vapor region ***
c
      mairs=mnis+moxs+mas+mco2s
       cpn2s=(0.172+8.57e-06*tgs+1.02e-09*tgs*tgs)
       cpo2s=(0.184+3.2e-06*tgs-1.36e04/tgs/tgs)
       cpco2s=0.29
      ua= (mnis*cpn2s+moxs*cpo2s+mas*cpas+mco2s*cpco2s)*tgs
      uv2= ua+uwv2*mwv2
      uvzer2=uv2
      ulzer2=ul2
      ns=3
      return
    3 continue
c
c************** calculate the emissivity of the air-vapor mixture ***
c
       if((ph2o2/14.7*chs/4.).le. 0.5)then
       ew1= 0.37*(ph2o2/14.7*chs/4.)**0.67
       else
       ew1= 0.28+0.11*alog(ph2o2/14.7*chs/4.)
```

```
        endif
        estair= 0.0476*(ph2o2+pas)*ew1
        if(estair.gt.(1.4*ew1))estair= 1.4*ew1
        emgs=emgs+estair
        if(emgs.gt.1.0)emgs=1.0
       if (emgs .le. 0.005) emgs=0.005
c
c ********* determining secondary air-vapor mixture temperature ***
      htcar2=mairs*cpa2
      if(mwv2.le.0.)go to 70
      lts(1)=.false.
      lts(2)=.false.
      fr=.true.
      ts(1)=491.0
      ts(2)=1940.0
      if(tgs.lt.ts(1))tgs=ts(1)
      vg2=(vs-vl2)/mwv2
 51   tgps=tgs
      cell=2.
      call proptv
 52   temp=uv2-htcar2*tgs-mwv2*uwv2
 55   j=2
      if(temp.gt.0.)j=1
      ts(j)=tgs
      lts(j)=.true.
      if(fr) go to 57
      if(abs(temp/uv2).le.0.0005)go to 56
      if(.not.(lts(1).and.lts(2)))go to 58
      if(abs(ts(1)-ts(2)).le.0.005)go to 56
 58   temp3= temp2-temp
      if(temp3.eq.0.)go to 1902
      temp4= deltat/temp3
      if(temp4.le.0.)go to 1902
      deltat=temp*temp4
 61   j=j+1
      if(j.eq.3)j=1
      temp3=ts(j)-tgs
      if(abs(deltat).le.abs(temp3))go to 60
      if(lts(j))go to 59
      deltat=temp3
      tgs=ts(j)
      go to 64
 59   deltat=0.9*temp3
 60   tgs=tgs+deltat
 64   temp2=temp
      go to 51
 57   fr=.false.
      deltat=sign(0.5,temp)
      go to 61
 56   if(phase2.eq.2.)xg2=1.0
      mwvv2=xg2*mwv2
      mwlv2=mwv2-mwvv2
      pas=ph2o2+mairs*rair2*tgs/(vs-vl2)/144.
      go to 1903
1902  deltat=0.5*(ts(1)+ts(2))-tgs
      go to 60
 70   ph2o2=0.0
      mwv2=0.0
      mwvv2=0.0
      mwlv2=0.0
      tgs=uv2/htcpgs
      pas=mairs*rair2*tgs/vs/144.
1903  continue
```

```
      ns=4
      return
    4 continue
c *********************************************
c******* check for boiling or condensation if water is present *******
      if(mwl2.gt.1.0e-06)then
       ulp2=ul2/mwl2
       cell=2.
       call proptu
c check for pool boiling
       do 1440 n=1,34
        p=n+1.
        if(pas.lt.sat(p,2))then
         intdsp=(pas-sat(n,2))/(sat(p,2)-sat(n,2))
         ulpb=(sat(p,5)-sat(n,5))*intdsp+sat(n,5)
         ugpb=(sat(p,6)-sat(n,6))*intdsp+sat(n,6)
         go to 444
        endif
 1440  continue
 444   continue
        boil=0.
       if(ulp2.gt.ulpb)then
        boil=1.
        mboil=(ul2-mwl2*ulpb)/(ugpb-ulpb)
        if(mboil.gt.mwl2)mboil=mwl2
        uvz2=uvz2+ugpb*mboil/delt
        mwlz2=mwlz2-mboil/delt
        mwvz2=mwvz2+mboil/delt
        ulz2=ulz2-ugpb*mboil/delt
       endif
       vl2=mwl2*vlp2
c
c without boiling, check for evaporation and condensation
c    q"=hb(tgs-tlp2)+[kb.18(hfg2+hf2)(mfvg-mfvb)]/xam
c    is the expression for heat transferred during evaporation
c
      if(boil.eq.0.)then
       mfvg=mwv2/(vs-vl2)/18./(mwv2/(vs-vl2)/18.+mairs/(vs-vl2)/xmols)
       if(mfvg.eq.1.0)mfvg=0.9999999
       mfag=1.-mfvg
c at the liquid-vapor boundary:
       mfvb=1./vvb2/18./(1./vvb2/18.+(pas-ph2ob2)*144./rair2/tlp2/xmols)
       if(mfvb.eq.1.0)mfvb=0.9999999
       mfab=1.-mfvb
       mflog=mfab/mfag
       if(mflog.eq.1.0)xam=1.0
       if(mflog.eq.1.0)go to 99
       xam=(mfab-mfag)/alog(mflog)
 99    continue
c determine h from the product GrPr
c find steam and air properties at the boundary and in the bulk vapor
       do 1430 n=1,2
        temp=abs(tlp2/1.8)
        if(n.eq.1)temp=abs(tgs/1.8)
        kair=.02665+6.013e-05*(temp-310.93)
        kair=kair/1.731/3600
        muair=1.912e-05+4.152e-08*(temp-310.93)
        muair=muair/1.488
        vvt=vvb2
        if(n.eq.1)vvt=vvg2
        kh2o=17.6+5.87e-02*(temp-273.)+1.04e-04*(temp-273.)**2
     .  -4.51e-08*(temp-273.)**3+(103.51+.4198*(temp-273.)-2.771e-05*
     .  (temp-273.)**2)*16.02/vvt*.001+2.148e08/(temp-273.)**4.2/
     .  (vvt/16.02)**2
        kh2o=kh2o*.001/1.729/3600.
        muv=11.4/(temp**2-884*temp+1.36e06)
        muv=muv/1.488
        if(n.eq.2)then
         kairb=kair
         muairb=muair
         kh2ob=kh2o
         muvb=muv
        else
         kairg=kair
         muairg=muair
         kh2og=kh2o
         muvg=muv
        endif
 1430 continue
       temp2=(muairb/muvb)**.5
       phiawb=.218942*(1.+.88808*temp2)**2
       phiwab=.277605*(1.+1.2603/temp2)**2
       vab=1.0e20
       if(pas.gt.ph2ob2)vab=1./rhoas
       vsb=vvb2*ph2ob2/pas
       mub=vsb*muairb/(vsb+vab*phiawb)+vab*muvb/(vab+vsb*phiwab)
       betab=1./tlp2
       rhob=1./vab+1./vsb
       cpb=(vsb*cpa2+vab*cpvb2)/(vab+vsb)
       kbndry=vsb*kairb/(vsb+vab*phiaw)+vab*kh2ob/(vab+vsb*phiwa)
       asurf=afs
c  calculate the heat transfer from GrPr
       grpr=betab*rhob*cpb*32.2*abs(tgs-tlp2)/kbndry*asurf**1.5/mub
       dab=2.478e-04*(14.22/pas)*(tgs/460.)**1.81
       if(tgs.gt.tlp2)then
        if(grpr.lt.2.e07)then
        ca=.54
        aa=.25
        else
        ca=.14
        aa=.3333
        endif
       elseif(tgs.lt.tlp2)then
        ca=.27
        aa=.25
       elseif((vvb2.eq.(vs-vl2)/mwvv2).and.(tgs.eq.tlp2))then
        go to 1999
       else
        ca=0.
        aa=1.0
       endif
c calculate the sensible heat transfer coefficient
       hsens=ca*kbndry/asurf**.5*grpr**aa
       mgb=18.*mfvb+xmols*mfab
       prsc=cpb*rhob*dab/kbndry
       kb=hsens/cpb/mgb*(prsc)**.666
c without a temperature difference, the mass transfer coefficient must be
c calculated differently than with one
       if(tgs.eq.tlp2) kb=1.02*pas*dab/(asurf**.5*1545.*tlp2)*
     .  (asurf**1.5*32.2*abs(mwvv2/(vs-vl2)-1./vvb2)/mub/dab)**.373
c
       a=kb*mgb*cpb*(mfvg-mfvb)/hsens/xam
c mass condensation/evaporation rate
       condr=kb*18.*(mfvg-mfvb)/xam*asurf
       if(condr.lt.(-1.*mwl2/delt))condr=-1.*mwl2/delt
c account for greater heat transfer with large mass transfer rates
```

```
       hprime=(a/(exp(a)-1.))*hsens
c heat transfer rate
       hcond=hprime*(tgs-tlp2)+condr/asurf*(hlp2+hfg2)
c transfer heat and mass to and from liquid and vapor regions
       mwvz2=mwvz2-condr
       mwlz2=mwlz2+condr
       uvz2=uvz2-hcond*asurf
       ulz2=ulz2+hcond*asurf
      endif
 1999 continue
      endif
c
c***********     calculating gas heat transfer coefficients    **********
c
c determine the effects of steam on the heat transfer coefficients
c find the saturation temperature and the Uchida heat transfer coeff.
      if(mwv2.gt.0.0)then
       do 1420 n=1,34
       p=n+1.
       if(ph2o2.lt.sat(p,2))then
       intdsp=(ph2o2-sat(n,2))/(sat(p,2)-sat(n,2))
       tsat=intdsp*(sat(p,1)-sat(n,1))+sat(n,1) +460.
       hsat=intdsp*(sat(p,7)-sat(n,7))+sat(n,7)
       go to 1111
       endif
 1420  continue
 1111  continue
       mr=mairs/mwv2
       do 1410 n=1,14
       p=n+1
       if(mr.gt.hu(p,1))then
       intdsh=(mr-hu(p,1))/(hu(n,1)-hu(p,1))
       huch= hu(p,2)-intdsh*(hu(p,2)-hu(n,2))
       go to 2222
       endif
 1410  continue
 2222  continue
       if(mr.ge.50.)huch=2.
       if(mr.le.0.1)huch=280.
       huch=huch/3600.
      endif
      ns=5
      return
    5 continue
c ***calculate heat transfer coefficients with steam present***
c
      if(mwv2.gt.0.0)then
       rhogs=rhoap+1./vg2
c  secondary gas to secondary wall
      qvsec=2.*(tsat-tss)*aws/(thws/kstlws+2./huch)
      if(tss.gt.tsat)then
       hsec=hingss*akexx(tgs,tss,rhogs)
       qvsec=2.*(tgs-tss)/(thws/kstlws+2./hsec)
      endif
      mcondw=qvsec/hfg2
      if(tss.gt.tsat .or. qvsec.le.0.0)mcondw=0.
      mwvz2=mwvz2-mcondw
      mwlz2=mwlz2+mcondw
      uvz2=uvz2-qvsec
      ulz2=mcondw*hsat+ulz2
      zzs=zzs+qvsec/rhsws/aws/thws/cpsws
c
c  secondary gas to primary wall
```

```
      qvwpgs=2.*(tsat-tsp)*awp/(thwp/kstlwp+2./huch)
      if(tsp.gt.tsat)then
       hwpgas=hinps*akexx(tsp,tgs,rhogs)
       qvwpgs=2.*(tgs-tsp)*awp/(thwp/kstlwp+2./hwpgas)
      endif
      mconwp=qvwpgs/hfg2
      if(tsp.gt.tsat .or. qvwpgs.le.0.0)mconwp=0.
      mwvz2=mwvz2-mconwp
      mwlz2=mwlz2+mconwp
      uvz2=uvz2-qvwpgs
      ulz2=mconwp*hsat+ulz2
      zz5=zz5+qvwpgs/rhswp/awp/thwp/cpswp
c
c  secondary gas to primary floor
      qvfpgs=2.*(tsat-tsfp)*afp/(thfp/kstlfp+1./huch)
      if(tsfp.gt.tsat)then
       hfpgas=hinfgs*akexx(tsfp,tgs,rhogs)
       qvfpgs=2.*(tgs-tsfp)*afp/(thfp/kstlfp+2./hfpgas)
      endif
      mconfp=qvfpgs/hfg2
      if(tsfp.gt.tsat .or. qvfpgs.le.0.0)mconfp=0.
      mwvz2=mwvz2-mconfp
      mwlz2=mwlz2+mconfp
      uvz2=uvz2-qvfpgs
      ulz2=mconfp*hsat+ulz2
      zz7=zz7+qvfpgs/rhsfp/afp/thfp/cpsfp
c
      endif
c
c*** liquid pool to secondary floor ******************
      if(mwl2.gt.1.0e-06)then
c determine average fluid properties
       tave=(tlp2+tfs)/2.
       tavhi=int(tave/20.)-23.
       tavlo=tavhi-1.
       intdsr=(tave/20.)-int(tave/20.)
       mulv=(25.3/((tave/1.8)**2.+91.*tave/1.8-8.58e04))/1.488
       cplv=(sat(tavhi,9)-sat(tavlo,9))*intdsr+sat(tavlo,9)
       vlpv=(sat(tavhi,3)-sat(tavlo,3))*intdsr+sat(tavlo,3)
       nulv=mulv*vlpv
       kwat=(0.686-5.87e-06*(abs(tave/1.8-415.))**2.+7.3e-10*pas*6895.)
       kwat=kwat/1.73/3600.
       if(tave.gt.1165.)then
        nulv=1.46e-06
        kwat=8.667e-05
        cplv=1.368
        vlpv=sat(34,3)
       endif
       tfhi=int(tfs/20.)-23.
       tflo=tfhi-1.
       intdsf=(tfs/20.)-int(tfs/20.)
       vlpf=(sat(tfhi,3)-sat(tflo,3))*intdsf+sat(tflo,3)
       if(tfs.gt.1165.)vlpf=sat(34,3)
       betaf=abs(1./vlp2*(vlpf-vlp2)/(tfs-tlp2))
c determine h from GrPr
       grprf=32.2*betaf*abs(tfs-tlp2)*afs**1.5
     . /vlpv/kwat*cplv/nulv
       if((tfs.gt.tlp2).and.(grprf.le.2.e07))then
        ca=0.54
        aa=0.25
       elseif((tfs.gt.tlp2).and.(2.0e07.lt.grprf).and.
     .        (grprf.le.3.0e10))then
        ca=0.14
        aa=0.333
       elseif(grprf.gt.3.0e10)then
        ca=0.021
        aa=0.4
       elseif((tfs.lt.tlp2).and.(grprf.le.3.e10)) then
        ca=0.27
        aa=0.25
       elseif(tlp2.eq.tfs)then
        ca=0.0
        aa=1.0
       endif
       hlpflr=kwat/afs**.5*ca*grprf**aa
       zh2o2=mwl2*vlp2/afs
       if(hlpflr.lt.(2.*kwat/zh2o2))hlpflr=2.*kwat/zh2o2
       qlflr=2.*(tlp2-tfs)/(thfs/kstlfs+1./hlpflr)
       ulz2=ulz2-qlflr
       zzfs=zzfs+qlflr/rhsfs/afs/thfs/cpsfs
c
      elseif(mwv2.gt.0.0 .and. mwl2.le.1.0e-06)then
c
c     secondary gas to secondary floor--no liquid water
       qvflr=2.*(tsat-tfs)*afs/(thfs/kstlfs+1./huch)
       if(tfs.gt.tsat)then
        hfsgas=hinfsg*akexx(tfs,tgs,rhogs)
        qvflr=2.*(tgs-tfs)*afs/(thfs/kstlfs+2./hfsgas)
       endif
       mcondf=qvflr/hfg2
       if(tfs.gt.tsat.or.qvflr.le.0.0)mcondf=0.0
       mwvz2=mwvz2-mcondf
       mwlz2=mwlz2+mcondf
       uvz2=uvz2-qvflr
       ulz2=ulz2+mcondf*hsat
       zzfs=zzfs+qvflr/rhsfs/afs/thfs/cpsfs
      endif
c
c secondary gas to secondary extraneous heat capacity
      if(mwv2.gt.0.0)then
c
      qvehc=huch*(tsat-tehcs)*aehcs
      if(tehcs.gt.tsat)then
       hehcs=hinecs*akexx(tgs,tehcs,rhogs)
       qvehc=(tgs-tehcs)*hehcs*aehcs
      endif
      mconde=qvehc/hfg2
      if(tehcs.gt.tsat .or. qvehc.le.0.0)mconde=0.
      mwvz2=mwvz2-mconde
      mwlz2=mwlz2+mconde
      uvz2=uvz2-qvehc
      ulz2=ulz2+mconde*hsat
      zzes=zzes+qvehc/xmehcs/cpehcs
c
      endif
c
      ns=3
      return
      end
c
c ***************************************

c
c this is the subroutine that creates the steam tables for the
c lithium-steam reaction option
c
```

```
      subroutine table
      implicit real (i,k,l,m)
      integer r
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .               ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .               phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +               mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .               xmolp,cell
c
c    the following data statements are used to define constants used
c    in determining the properties of water and steam
      data tsc1,tsc2,tsexp  /9.0395, 255.2, 0.223/
      data cps1,cps2,cpsexp  /9.5875e02, 0.00132334, -0.8566/
      data g11,g12,g13  /2.6194106e06, -4.995e10, 3.403e05/
      data g14,g15,g16  /1.0665544, 1.02e-08, -2.548e-15/
      data g17  /-5.096e-15/
      data hl0,hl1,hl2,hl3,hl4,hl5  /5.7474718e05, 2.0920624e-01,
     .    -2.8051070e-08, 2.3809828e-15, -1.0042660e-22, 1.6586960e-30/
      data hv0,hv1,hv2,hv3,hv4  /2.7396234e06, 3.758844e-02,
     .     -7.1639909e-09, 4.2002319e-16, -9.8507521e-24/
      data tcrit,tcrinv  /647.3, .00154488/
      data cc,cci,ccm  /1.3, .76923, 0.3/
      data rl0,rl1,rl2,rl3  /1735.3320, -4.6406842, 1.0431090e-02,
     .     -9.4367085e-06/
      data rh0,rh1,rh2,rh3,rh4  /-1.1755984e06, 8.1437361e03,
     .     -21.136559, 2.4381598e-02, -1.0549747e-05/
      data rp0,rp1,rp2  /-14.643890, 1.1283357e-03, 1.2670366e-02/
      data sp0,sp1,sp2,sp3  /-42.0218, 0.2116, -4.4587e-04, 3.251e-07/
      data sp22,sp33  /-8.9174e-04, 9.753e-07/
      data sl0,sl1,sl2,sl3,sl4  /-460268.18, -2863.4045, 27.450693,
     .     -4.8108323e-02, 3.2059316e-05/
      data sl22,sl33,sl44  /54.901386, -.14432497, 1.2823726e-04/
      data sh0,sh1,sh2,sh3,sh4  /1.2426455e09, -8608225.1, 22364.564,
     .     -25.815959, 1.1178766e-02/
      data sh22,sh33,sh44  /44729.128, -77.447877, 4.4715064e-02/
      data a11,a12,a13  /1.2959e-03, 593.59, 1.6847e-03/
c
c    create the array "sat(35,10)" for saturated properties
c
      t=277.6
      do 1400 n=1,35
      p=((t-tsc2)/tsc1)**(1./tsexp)
      sat(n,2)=p/6895.
c    find the saturated enthalpies
      hg= hv0+p*(hv1+p*(hv2+p*(hv3+p*hv4)))
      hf= hl0+p*(hl1+p*(hl2+p*(hl3+p*(hl4+p*hl5))))
      sat(n,7)= hf/2321.
      sat(n,8)= hg/2321.
c   find the saturated internal energy
      t1= 1.-t*tcrinv
      if(t1.lt.0.0)t1=1.0e-10
      cps= cps1*(t1**cpsexp)
      t2= 1./(g13+p)
      t1= t2*g12
      es= g11+t1
      sat(n,6)= es/2321.
      gams= g14+p*(g15+p*g16)
      gamsm= gams-1.
      dp= p-1.5e07
      deldp= -exp(sp0+t*(sp1+t*(sp2+t*sp3)))
      del= deldp*dp
      ul= sl0+t*(sl1+t*(sl2+t*(sl3+t*sl4)))+del
      if(t.ge.576.5)ul= sh0+t*(sh1+t*(sh2+t*(sh3+t*sh4)))+del
      sat(n,5)= ul/2321.
c  find the specific heat at constant volume
      deldt= sl1+t*(sl22+t*(sl33+t*sl44)) + del*(sp1+t*(sp22+t*sp33))
      if(t.ge.576.5)deldt= sh1+t*(sh22+t*(sh33+t*sh44)) + del*(sp1+
     .   t*(sp22+t*sp33))
      devdt= cps*cci
      sat(n,9)= deldt/4187.
      sat(n,10)= devdt/4187.
c  find the specific volumes
      drldp= exp(rp0+rp1*exp(rp2*t))
      drl= drldp*dp
      rol= rl0+ t*(rl1+t*(rl2+t*rl3))+ drl
      if(t.ge.576.5)rol= rho+ t*(rh1+t*(rh2+t*(rh3+t*rh4))) +drl
      sat(n,3)= 1./(rol/16.02)
      ev= es
      rov= p/gamsm/ev
      sat(n,4)= 1./(rov/16.02)
      sat(n,1)= t*1.8-460
      t=t+11.11
 1400 continue
c
c  these data statements are necessary to correct inaccuracies in the
c  steam table formulae used in the subroutine
      sat(1,2)=0.1217
      sat(2,2)=0.2563
      sat(3,2)=0.5073
      sat(4,2)=0.9503
      sat(5,2)=1.695
      sat(6,2)=2.892
      sat(7,2)=4.745
      sat(8,2)=7.515
c spec. vol. sat. liq.
      sat(28,3)=.02278
      sat(29,3)=.02363
      sat(30,3)=.02465
      sat(31,3)=.02593
      sat(32,3)=.02767
      sat(33,3)=.03032
      sat(34,3)=.03666
      sat(35,3)=.090
c spec. vol. sat. vap.
      sat(1,4)=2445.
      sat(2,4)=1207.
      sat(3,4)=632.8
      sat(4,4)=350.0
      sat(5,4)=203.0
      sat(6,4)=122.9
      sat(7,4)=77.2
      sat(8,4)=50.2
      sat(23,4)=.8187
      sat(24,4)=.6761
      sat(25,4)=.5605
      sat(26,4)=.4658
      sat(27,4)=.3877
      sat(28,4)=.3225
      sat(29,4)=.2677
      sat(30,4)=.2209
      sat(31,4)=.1805
      sat(32,4)=.1446
      sat(33,4)=.1113
      sat(34,4)=.0744
      sat(35,4)=-.0140
c spec. enth. sat. liq.
```

```
      sat(1,7)=8.02
      sat(2,7)=28.08
      sat(3,7)=48.09
      sat(4,7)=68.05
      sat(5,7)=88.0
      sat(6,7)=107.96
      sat(7,7)=127.96
      sat(8,7)=147.99
      sat(9,7)=168.07
      sat(10,7)=188.2
      sat(11,7)=208.4
      sat(12,7)=228.8
      sat(13,7)=249.2
      sat(35,7)=1118.3
c spec. enthalpy sat vap.
      sat(1,8)=1078.9
      sat(2,8)=1087.7
      sat(3,8)=1096.4
      sat(4,8)=1105.0
      sat(5,8)=1113.5
      sat(6,8)=1121.9
      sat(7,8)=1130.1
      sat(8,8)=1138.2
      sat(9,8)=1145.9
      sat(34,8)=990.2
      sat(35,8)=665.4
c spec. energy sat liq.
      sat(1,5)=8.02
      sat(34,5)=801.7
      sat(35,5)=1064.3
c spec. energy sat. vap.
      sat(1,6)=1023.9
      sat(2,6)=1030.4
      sat(3,6)=1037.0
      sat(4,6)=1043.5
      sat(5,6)=1049.9
      sat(6,6)=1056.2
      sat(7,6)=1062.3
      sat(8,6)=1068.3
      sat(28,6)=1098.9
      sat(29,6)=1090.0
      sat(30,6)=1078.5
      sat(31,6)=1063.2
      sat(32,6)=1042.3
      sat(33,6)=1011.0
      sat(34,6)=947.7
      sat(35,6)=669.6
c
c  create the array "sh(7,70,5)" for superheated steam properties
      p= 6895.
      do 1390 n=1,7
      t=310.9
      tsat= tsc1*p**tsexp
      tsat= tsat+tsc2
      t1= 1.-tsat*tcrinv
      cps= cps1*t1**cpsexp
      t2= 1./(g13+p)
      t1= t2*g12
      es= g11+t1
      gams= g14+p*(g15+p*g16)
      gamsm= gams-1.
      t1= 1./(a11*cps-1.)
      beta= tsat**2*(1.-t1**2)
      t2= tsat*t1
       do 1380 r=1,70
c   find the specific internal energy
       dt= t-tsat
       t3=abs(t**2-beta)
       de= a12*(dt+sqrt(t3)-t2)
       ev=es+de
       sh(n,r,3)= ev/2321.
       if(n.eq.1 .and. r.le.13)sh(n,r,3)=sh(n,r,3)*.986
c   find specific volume
       t4=1./(gamsm*es+ccm*de)
       rov=p*t4
       sh(n,r,2)= 1./(rov/16.02)
       if(n.eq.1)sh(n,r,2)=sh(n,r,2)*1.05
c   find specific enthalpy
       sh(n,r,4)= (ev+p/rov)/2321.
       sh(n,r,1)= t*1.8-460.
c   find specific heat at constant volume
       sh(n,r,5)= cps*cci/4187.
       t= t+11.11
 1380 continue
      if(n.eq.1)p=p+27580.
      if(n.eq.2)p=p+34475.
      if(n.ge.3)p=p+68950.
 1390 continue
c
c  these data statements are necessary to correct inaccuracies
c  in the formulae used to create the steam table
c
      sh(1,1,2)=333.6
      sh(1,2,2)=345.2
      sh(1,3,2)=356.8
      sh(1,4,2)=368.6
      sh(1,5,2)=380.5
      sh(1,6,2)=392.5
      sh(1,7,2)=404.5
      sh(1,8,2)=416.4
      sh(1,9,2)=428.4
      sh(1,10,2)=440.3
      sh(1,11,2)=452.3
      sh(1,12,2)=464.2
      sh(1,13,2)=476.1
      sh(1,14,2)=488.1
      sh(1,15,2)=500.0
      sh(1,16,2)=511.9
      sh(1,17,2)=523.8
      sh(1,18,2)=535.7
      sh(1,19,2)=547.7
      sh(1,20,2)=559.5
      sh(1,21,2)=571.5
c
c  create a table of Uchida heat transfer coefficients for
c  steam condensation. Table "hu(15,2)"
c
      data hu(1,1),hu(2,1),hu(3,1),hu(4,1),hu(5,1),hu(6,1),
     .     hu(7,1),hu(8,1),hu(9,1),hu(10,1),hu(11,1),
     .     hu(12,1),hu(13,1),hu(14,1),hu(15,1) /50., 20.,
     .     18., 14., 10., 7., 5., 4., 3.0, 2.3, 1.8, 1.3, 0.8, 0.5,0.1/
      data hu(1,2),hu(2,2),hu(3,2),hu(4,2),hu(5,2),hu(6,2),
     .     hu(7,2),hu(8,2),hu(9,2),hu(10,2),hu(11,2),
     .     hu(12,2),hu(13,2),hu(14,2),hu(15,2) /2., 8., 9.,
     .     10., 14., 17., 21., 24., 29., 37., 46., 63., 98., 140.,280./
      return
      end
c
c
c these are the subroutines used to determine the properties of water
c in the containment, using the table created in the previous subroutine.
c
      subroutine proptv
      implicit real (i,k,l,m)
      integer k,p,phi,plo,q,r,thi,this,tlo,tlos
      dimension tabtmp(10),proprt(10)
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .               ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .               phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +               mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .               xmolp,cell
      common /steam2/cpa2,cpvb2,emgs,hfg2,hlp2,htcpgs,hum2,mwl2,
     .               mwlv2,mwlz2,mwlzr2,mwv2,mwvv2,mwvz2,mwvzr2,
     .               phase2,ph2o2,ph2ob2,rair2,tlp2,ul2,ulp2,ulz2,
     .               ulzer2,uv2,uvz2,uvzer2,uwv2,vg2,vl2,vlp2,vvb2,
     .               vvg2,xg2,xmols,stmin2,stout2,xinj2,rhoas
c
c  determine whether the mixture is saturated or superheated
c   find the temperatures between which to interpolate
      if(cell.eq.1.)vgs=vg
      if(cell.eq.2.)vgs=vg2
      tlo= int(tgps/20.)-24.
      thi= tlo+1.
      intdis= tgps/20.-int(tgps/20.)
c   check to see if the mixture is saturated or superheated
      vvg= (sat(thi,4)-sat(tlo,4))*intdis+ sat(tlo,4)
      if(vgs.gt.vvg .or. tgps.gt.1165.)then
      phases=2.
c   mixture is superheated
      tlos= tlo-3
      this= tlos+1
c    check to see if the properties fit on the chart
        vcheck=.5937*tgps
        if(tlos.le.0.0 .or. vgs.gt.vcheck)then
         ph2os=1545./18.*tgps/vgs/144.
         cpwvs=sh(1,this,5)
         if(tlos.le.0.0)then
          uwvs=1044.0-16.4*(560.-tgps)/48.3
          hwvs=1105.8-21.7*(560.-tgps)/48.3
         else
          uwvs=(sh(1,this,3)-sh(1,tlos,3))*intdis+sh(1,tlos,3)
          hwvs=(sh(1,this,4)-sh(1,tlos,4))*intdis+sh(1,tlos,4)
         endif
         go to 500
        endif
c
      do 1370 n=1,6
       r=n+1
       vlo=(sh(n,this,2)-sh(n,tlos,2))*intdis +sh(n,tlos,2)
       vhi=(sh(r,this,2)-sh(r,tlos,2))*intdis +sh(r,tlos,2)
       if(vhi.lt.vgs)then
        phi=r
        plo=n
        intdsv=(1./vgs-1./vlo)/(1./vhi-1./vlo)
        go to 100
       endif
 1370 continue
 100  continue
c  determine the water vapor partial pressure
```

```
      if(n.eq.1)ph2os= 1.+4.*intdsv
      if(n.eq.2)ph2os= 5.+5.*intdsv
      if(n.ge.3)ph2os= (plo-2.+ intdsv)*10.
c  determine the other properties
      do 1360 j=3,5
      k=j+1
      proprt(k)=((sh(phi,this,j)-sh(phi,tlos,j))*intdis +sh(phi,tlos,j)
     .         -(sh(plo,this,j)-sh(plo,tlos,j))*intdis +sh(plo,tlos,j))
     .         *intdsv+ (sh(plo,this,j)-sh(plo,tlos,j))*intdis
     .         +sh(plo,tlos,j)
 1360 continue
      else
c   mixture is saturated
      phases= 1.
      do 1350 n=1,10
      tabtmp(n)= (sat(thi,n)-sat(tlo,n))*intdis+ sat(tlo,n)
 1350 continue
c   find mixture quality
      xgs= (vgs-tabtmp(3))/(tabtmp(4)-tabtmp(3))
c   find the mixture properties
      do 1340 n=4,6
      p= 2*(n-1)
      q= p-1
      proprt(n)= (tabtmp(p)-tabtmp(q))*xgs+ tabtmp(q)
 1340 continue
      endif
c  assign properties
      if(phases.eq.1.)ph2os=tabtmp(2)
      uwvs=proprt(4)
      hwvs=proprt(5)
      cpwvs=proprt(6)
      vvgs=tabtmp(4)
 500   continue
      hfgs=(sat(thi,8)-sat(tlo,8))*intdis+sat(tlo,8)-(sat(thi,7)-
     .    sat(tlo,7))*intdis-sat(tlo,7)
      if(phases.eq.2.)vvgs=vgs
c assign primary cell properties
      if(cell.eq.1.)then
       ph2o=ph2os
       uwv=uwvs
       hwv=hwvs
       cpwv=cpwvs
       vvg=vvgs
       hfg=hfgs
       xg=xgs
       phase=phases
      endif
c assign secondary cell properties if needed
      if(cell.eq.2.)then
       ph2o2=ph2os
       uwv2=uwvs
       hwv2=hwvs
       cpwv2=cpwvs
       vvg2=vvgs
       hfg2=hfgs
       xg2=xgs
       phase2=phases
      endif
      return
      end
c
c this subroutine is used to determine the properties of liquid water
c
```

```
      subroutine proptu
      implicit real (i,k,l,m)
      integer p,q
      dimension proprt(10)
      common /steam/ tgps,vg,xg,sat(35,10),sh(7,70,5),uwv,hwv,cpwv,vvg,
     .               ph2o,ulp,tlp,vlp,hlp,vvb,hfg,ph2ob,cpvb,hu(15,2),
     .               phase,hum,cpa,mwl,mwv,ul,uv,ulz,uvz,mwlz,
     +               mwvz,uvzer,ulzer,mwvv,mwlv,vl,zzep,mwvzer,mwlzer,
     .               xmolp,cell
      common /steam2/cpa2,cpvb2,emgs,hfg2,hlp2,htcpgs,hum2,mwl2,
     .               mwlv2,mwlz2,mwlzr2,mwv2,mwvv2,mwvz2,mwvzr2,
     .               phase2,ph2o2,ph2ob2,rair2,tlp2,ul2,ulp2,ulz2,
     .               ulzer2,uv2,uvz2,uvzer2,uwv2,vg2,vl2,vlp2,vvb2,
     .               vvg2,xg2,xmols,stmin2,stout2,xinj2,rhoas
c
      if(cell.eq.2.)ulps=ulp2
      if(cell.eq.1.)ulps=ulp
      do 1330 n=1,34
      p=n+1.
      if(ulps.lt.sat(p,5))then
      intdsu=(ulps-sat(n,5))/(sat(p,5)-sat(n,5))
      go to 333
      endif
 1330 continue
 333  continue
      do 1320 q=1,10
      proprt(q)=(sat(p,q)-sat(n,q))*intdsu+sat(n,q)
 1320 continue
      tlps=proprt(1)+460.
      vlps=proprt(3)
      hlps=proprt(7)
c   boundary layer properties
      vvbs=proprt(4)
      ph2obs=proprt(2)
      cpvbs=proprt(10)
      hfgs=proprt(8)-proprt(7)
c convert to primary cell properties
      if(cell.eq.1.)then
       tlp=tlps
       vlp=vlps
       hlp=hlps
       vvb=vvbs
       ph2ob=ph2obs
       cpvb=cpvbs
       hfg=hfgs
      endif
c convert to secondary cell properties
      if(cell.eq.2.)then
       tlp2=tlps
       vlp2=vlps
       hlp2=hlps
       vvb2=vvbs
       ph2ob2=ph2obs
       cpvb2=cpvbs
       hfg2=hfgs
      endif
      return
      end
c
c
c this is the system international unit conversion subroutine allowing
c the input and output to be prepeared and written in si units.
      subroutine si
      implicit real (i,k,l,m)
      logical flagw,flagf,flag2,flagpn,flagc,flagas,flagn,flagdf,flagst
      common // name(340),flag2,flagas,flagc,flagf,flagn,flagst,
     .         flagpn,flagw,ipage,iswich,iarosl,flagdf,icz,flagco
      common /looper/ i10,i11,i12,i13,i14,i15,i16,i17,i18,i19
      common /lith/ akli,asli,cpli,csbli,hb,libp,lil,lilp,lit,
     .              rhli,spill,tli,tlii,zli
      common /steel/ cpsfp,cpsfs,cpswp,cpsws,estlfp,estlwp,kstlfp,
     .               kstlfs,kstlwp,kstlws,rhsfp,rhsfs,rhswp,rhsws
      common /misc/ afp,afs,awp,aws,c7,c21,gin,
     .              ha,hinfam,hinsam,htcpgp,qradc,radc,rczw,
     .              rhoap,rliw,rwpws,sigma,ta,tc(20),tfs,
     .              tfszer,tgp,tgs,tgpzer,tsfp,tsp,tss,
     .              tsszer,thfp,thfs,thwp,thws,zzes,zz5,zzs,zz1,zz7,
     .              rair
      common /injop/ dp1,dp2,dp3,mniinj,moxinj,time,vp
      common /panop/ ains,apan,bredth,clist,cpins,cppan,emgp,fpg,fpw,
     .               kpan,rhins,rhpan,thkin1,thkin2,thkpan,
     .               tins1,tins1f,tins1i,tins2,tins2f,tins2i,
     .               tpan,tpanf,tpanzo,zz2,zz4,zz8,zz9
      common /conop/ c8,cpcon,dtbdt(20),dtcdt(20),gap,kcon,kgap,
     .               l(20),l1(20),nl,nl1,qradb,radb,rhcon,
     .               sflcr,tb(20),tbf(20),tbic(20),tcf(20),
     .               tcic(20),thfc,thwc,tsfpi,tspzer,xsfl,qlflr,qvflr
      common /ccop/ cmbro,cracon,dcocz,h2left,qcconc,rcmbo,rcmbw,
     .              relese,tcigni,tcon,tconf,xmh2oi,zzc,zzd,zzdin
      common /pbpool/ dmpbdt,zzpb,mlead,tleadi,xwli,dflipb,xlidot,
     .                thpb,tleadf,foo
      common /pbdif/ cczp,cgli,clig,cpcz,cpmcz,dfilm,kfilm,pyup,
     .               qradp,rczp,rgli,rifczp,rifpg,rifpw,rlig,rwli,
     .               tlead,yapcz,zz6
      common /secop/ aehcs,c11,c20,chs,cpehcs,cph2,cplih,cpwa,crack,
     .               foutp,fouts,foutt,hinfgs,hinfsg,hingss,hinps,kleak,
     .               leak,mairp,mairs,mais,mas,mh2s,mlihs,mlinis,mlins,
     .               mliois,mlios,mniis,mnis,moxis,moxs,mwais,
     .               mlc3s,mlc2s,mcs,mco2s,rholc3,rholc2,rhocar,
     .               mwas,pap,pas,paszer,ra,rbreak,rholih,
     .               rholin,rholio,rwpgas,tehcs,tehcsf,tehczs,tgsf,
     .          tfsf,tgszer,tssf,vs,xmdot,xmehcs,xmola,zz3,zzfs
      common /units/ aehcp,beta,chp,cmbrh,cpap,cpehcp,map,mnip,
     .               moxp,mwap,papzer,qcn,qco,qco1,qco2,qcw,qvap,
     .               tcz,tczf,tczi,tehcp,tehcpf,tehczp,tgpf,
     .               tlif,tmelt,tsfpf,tspf,tvap,xmehcp
      common /stmop/ minjr,minjr2,hinj,hinj2
c
      if (flagn) n=2
      go to (1,2,3)n
    1 continue
      aehcp=aehcp*10.765
      afp=afp*10.765
      akli=akli*0.57803
      asli=asli*10.765
      awp=awp*10.765
      chp=chp*3.281
      cpap=cpap*2.389e-04
      cpcon=cpcon*2.389e-04
      cpehcp=cpehcp*2.389e-04
      cpli=cpli*2.389e-04
      cpsfp=cpsfp*2.389e-04
      cpswp=cpswp*2.389e-04
      gap=gap*3.281
      kleak=kleak*0.03771
      kcon=kcon*0.57803
      kgap=kgap*0.57803
      kstlfp=kstlfp*0.57803
      kstlwp=kstlwp*0.57803
      papzer=papzer*1.450e-01
      qcn=qcn*4.311e-01
      qco=qco*4.311e-01
      qco1=qco1*4.311e-01
      qco2=qco2*4.311e-01
      qcw=qcw*4.311e-01
      qvap=qvap*4.311e-01
      rhcon=rhcon*0.062428
      rhli=rhli*0.062428
      rholih=rholih*0.062428
      rholin=rholin*0.062428
      rholio=rholio*0.062428
      rhsfp=rhsfp*0.062428
      rhswp=rhswp*0.062428
      spill=spill*2.2046
      ta=ta*1.8
      tczi=tczi*1.8
      tehczp=tehczp*1.8
      tgpzer=tgpzer*1.8
      thfc=thfc*3.281
      thwc=thwc*3.281
      thfp=thfp*3.281
      thwp=thwp*3.281
      tlii=tlii*1.8
      tmelt=tmelt*1.8
      tsfpi=tsfpi*1.8
      tspzer=tspzer*1.8
      tvap=tvap*1.8
      vp=vp*35.32
      xmehcp=xmehcp*2.2046
      zli=zli*3.281
c
      if (flagst) then
       minjr=minjr*2.2046
       minjr2=minjr2*2.2046
       hinj=hinj/2.326
       hinj2=hinj2/2.326
      endif
c
      if (.not. flag2) go to 100
      aehcs=aehcs*10.765
      afs=afs*10.765
      aws=aws*10.765
      chs=chs*3.281
      cpas=cpas*2.389e-04
      cpehcs=cpehcs*2.389e-04
      cpsfs=cpsfs*2.389e-04
      cpsws=cpsws*2.389e-04
      crack=crack*0.1550
      kstlfs=kstlfs*0.57803
      kstlws=kstlws*0.57803
      paszer=paszer*1.450e-01
      rhsfs=rhsfs*0.062428
      rhsws=rhsws*0.062428
      tehczs=tehczs*1.8
      tfszer=tfszer*1.8
      tgszer=tgszer*1.8
      thfs=thfs*3.281
      thws=thws*3.281
      tsszer=tsszer*1.8
      vs=vs*35.32
       xmehcs=xmehcs*2.2046
  100 continue
c
       if (.not. flagpn) go to 101
       ains=ains*10.765
       apan=apan*10.765
       bredth=bredth*3.281
       cpins=cpins*2.389e-04
       cppan=cppan*2.389e-04
       kpan=kpan*0.57803
       rhins=rhins*0.062428
       rhpan=rhpan*0.062428
       thkin1=thkin1*3.281
       thkin2=thkin2*3.281
       thkpan=thkpan*3.281
       tpanzo=tpanzo*1.8
  101 continue
c
       if (iblow .ne. 1) go to 102
       blowv=blowv*2119.2
       cpab=cpab*2.389e-04
       exhstv=exhstv*2119.2
       tblow=tblow*1.8
  102 continue
       if (isflc .eq. 1) sflcr=sflcr*9.475e-04
       if (iesc .eq. 1) escr=escr*9.475e-04
c
       if (iarosl .eq. 1) beta=beta/3.281
c
       if (.not. flagc) go to 103
       cracon=cracon*10.765
       qcconc=qcconc*4.311e-01
       tcigni=tcigni*1.8
       xmh2oi=xmh2oi*2.2046
       zzdin=zzdin*3.281
  103 continue
c
       if (.not. flagas) go to 104
       dp1=dp1*1.450e-01
       dp2=dp2*1.450e-01
       dp3=dp3*1.450e-01
  104 continue
    2 continue
    3 continue
       return
       end
```

References

1. D.A. Dube and M.S. Kazimi, "Analysis of Design Strategies for Mitigating the Consequences of Lithium Fire Within Containment of Controlled Thermonuclear Reactors", MITNE-219, July, 1978.
2. M.S. Tillack and M.S. Kazimi, "Development and Verification of the LITFIRE code for Predicting the Effects of Lithium Spills in Fusion Reactor Containments", MIT PFC/RR-80-11, July 1980.
3. V. Gilberti and M.S. Kazimi, "Lithium Fire Modeling in Multi-Compartment Systems", MIT PFC/RR-83-08, January 1983.
4. D.S. Barnett and M.S. Kazimi, "Chemical Kinetics of the Reactions of Lithium with Steam-Air Mixtures", in progress.
5. T.K. Gil and M.S. Kazimi, "The Kinetics of Liquid Lithium Reaction with Oxygen-Nitrogen Mixtures", PFC/RR-86-1, Plasma Fusion Center, MIT, January, 1986