United States Department of Commerce
National Institute of Standards and Technology

NIST Technical Note 1283

A Programmer's Reference Manual for CFAST, the Unified Model of Fire Growth and Smoke Transport

Walter W. Jones and Glenn P. Forney

NIST Technical Note 1283

A Programmer's Reference Manual for CFAST, the Unified Model of Fire Growth and Smoke Transport

Walter W. Jones
Glenn P. Forney

Center for Fire Research
National Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

November 1990

U.S. Department of Commerce
Robert A. Mosbacher, Secretary

National Institute of Standards and Technology
John W. Lyons, Director

National Institute of Standards
and Technology
Technical Note 1283
Natl. Inst. Stand. Technol.
Tech. Note 1283
104 pages (Nov. 1990)
CODEN: NTNOEF

U.S. Government Printing Office
Washington: 1990

For sale by the Superintendent
of Documents
U.S. Government Printing Office
Washington, DC 20402

CONTENTS

LIST OF FIGURES

A Programmer's Reference Manual for CFAST, the Unified Model of Fire Growth and Smoke Transport

Walter W. Jones
and
Glenn P. Forney

Center for Fire Research
National Institute of Standards and Technology

This document describes the unified model of fire growth and smoke spread, CFAST. This paper documents the internal structure of the model and details the method of modifying the model, together with examples. The intent is to provide a framework and methodology for maintenance of the model, together with a method of updating it. The reader is assumed to have a working knowledge of programming, software maintenance and modeling of physical phenomena.

1. INTRODUCTION

CFAST is the model that will be used by the Center for Fire Research for applications involving predictions of fire growth and smoke transport. It is not yet in the Hazard Methodology suite, but has been designated for release two. This manual describes the basis and philosophy behind the software implementation of CFAST. The name CFAST is used in two different ways, as a suite of programs and as the program for modeling fire and smoke transport. The primary programs in the CFAST suite are the data editor CF_IN for entering data, CFAST for modeling fire and smoke transport, CF_PLT for displaying plots of data, and CF_RPT for displaying reports.

This manual also describes procedures for updating CFAST. These procedures were developed to allow several people to work simultaneously and cooperatively on the model. The authors have tested this concept by merging changes each made into one version. The primary goal is to describe the philosopy behind the development of the unified model and detail the information that is crucial to making changes in the model. The internal structure

is from the FAST project, as are the physical routines. Thus this information is available in the references and is not detailed here.

CFAST is the result of a merger of ideas that came out of the CCFM.VENTS [1] and the FAST[2] development projects. Some of the issues addressed in the consolidation of FAST and CCFM.VENTS are discussed in section 1.1. The organization of the FAST suite of programs illustrated later in figure 3 remains the same in CFAST. However, the internal structure of these programs has been modified to incorporate the concepts used by CCFM.VENTS.

CFAST is a zone model, and as such there are limitations to its applicability. Within this context, the model can be revised, improved, and moved to other platforms than the one on which it was originally developed. Presently, CFAST runs on MSDOS[1] compatible computers. There are several incentives to "port" it to other platforms. First, programs that run on several platforms tend to be more robust. This is due to the fact that subtle errors in a program missed by one compiler are often caught by another. Second, other platforms such as UNIX workstations, are generally much faster than most MSDOS compatible computers. The main stumbling block to porting CFAST to other platforms is the hardware dependent nature of graphics libraries. DEVICE, the graphics library used by CFAST to display graphical representations of room fires supports several display devices but at present does not include the Apple Macintosh or any UNIX workstation. A test version of DEVICE has been enhanced to support X11[2]. By having an X11 driver, DEVICE will run on any UNIX workstation that supports X11. This includes most UNIX workstations on the market today. Additions for the Macintosh and other platforms are in the planning stage.

CFAST is not a final product, and as such the structure of the model, the modules associated with it, and the procedure for modification will change. The overall philosophy of its design and updating procedures should remain intact. It is intended that substantial revisions along these lines would be accompanied by appropriate revision documents.

In most cases it will be modified and used on a single computer. However, the intent of the structures described in this document is to provide a flexible environment for cooperative research by a group working on the model on a network. So the instructions will be given as if a single user were present, but the *same* statements will apply to the network environment as well. The differences will show up in the make files[3], where explicit

---

1 The use of company names or trade names within this manual is made only for the purpose of identifying computer hardware or software products. Such use does not constitute any endorsement of those products by the National Institute of Standards and Technology.

2 X11 is a vendor independent method for displaying screen graphics. This protocol allows one to compute graphical displays on one computer and display them on another.

3 A make file contains a collection of rules that are used by the tool MAKE to rebuild a software project. Using MAKE allows one to update software efficiently since only the subroutines that actually change need to be re-compiled.

references will occur to the network as well as the local environment. Since it has been tested in both environments, it is fair to say that the procedures work.

CFAST is a consolidation of the model FAST, and the concepts behind the CCFM work. These programs work on a variety of computers from basic PC's to super computers. Clearly the numerical portions run significantly faster on the latter than on the former. The only real restriction is that the platform must support a standard FORTRAN compiler, and basic graphics concepts. The methodology described in this document is aimed at maintaining a relatively sophisticated model which includes the phenomena now extant, plus algorithms under development, as well as phenomena which we can envision in the future. As an evolutionary product, the computer requirements will increase over time. Although not an explicit requirement at this time, it is the authors' intentions to require an unsegmented memory scheme after the first release of this model. Such a requirement precludes some computer platforms that have been usable in the past. However, it is a relatively simple matter, given the structure of the model and protocol, to trim the kernel so that it will once again operate on such platforms.

1.1 The Consolidation Process

The ideas and structures behind the design of the CCFM.VENTS software were examined so that CFAST could take advantage of the lessons learned during CCFM.VENTS' development. The basic thrust of most of these ideas was to promote flexibility of the software design. For example, a programming style was defined so that a program's precision could be changed. Also, rules for accessing global or common data were set up to make it easier to install new physical algorithms. Some of these issues addressed in the CCFM.VENTS, FAST consolidation process are discussed in more detail below.

common block access Low level physical routines should not have access to the global data structures contained in labeled common blocks. All data that is used or set by these routines nominally should be passed through the subroutine's argument list. CCFM.VENTS used this technique, FAST does not. It was not practical to incorporate such a scheme into CFAST. Instead, each physical routine was modified so that it could only reference, but not modify, common block variables. A tool, **comcheck** [3], was developed to identify common block variables that a subroutine changes. The common blocks in CFAST now serve as a consistent environment available to all routines.

variable precision CCFM.VENTS was coded using a programming style that allowed for easy conversion between single and double precision. FAST was modified by incorporating this style. CFAST can now be switched between single and double precision. This involves changing two lines in a make file and re-building the program. A **make** program, along with other development tools for generating versions of CFAST are discussed in sections 3.1-3.3. A few simple rules for writing software outlined in section 3.4 need to be followed in order to switch precision. A complication of this conversion was the necessity of

keeping certain routines in single precision since they accessed single precision graphics library routines that could not be converted to double precision.

software tools Software tools to find text differences and to examine program structure were developed during the CCFM.VENTS project. These programs were enhanced during the CCFM.VENTS/FAST consolidation project. The difference program, **fdiff**, was modified to make use of the fact that it was examining FORTRAN programs. The program structure tool, **roadmap**, was enhanced to work in an MSDOS environment. Another tool, **comcheck**, was developed to examine data structures found in common blocks. These tools are documented in reference [3].

data copy The solver arrays used in CCFM.VENTS and FAST have a well defined structure. For example for the first room, in CCFM.VENTS Y(1) = relative pressure, Y(2) = layer height, Y(3) = upper layer mass, Y(4) = lower layer mass. An analogous array, P, exists in CFAST. The 'Y' array is not used directly by physical routines in CCFM.VENTS. Rather a *data copy* is invoked that transfers information from the array Y to the environment variables that the physical routines access. This process of copying data makes it easier to change the formulation of the predictive equations, since all required changes can be made in a few routines. A *data copy* procedure also allows one to isolate further the physical routines from the data structures. The concept has been incorporated in CFAST. The control programs SOLVE and DSOURC invoke the procedure which make the environment available through the CENVIRO include file.

report generation CCFM.VENTS had a scheme for producing output reports from a history file. FAST did not have such a module. A routine CF_RPT has been incorporated into CFAST. The purpose of it is to allow output from a simulation to be produced from an archived history file without having to re-run the model that produced the original calculation.

history file format FAST stores the entire Common Block whenever an entry in the history file is produced. CCFM.VENTS stores a few setup variables at the beginning and the "solver" variables whenever an entry in the history file entry is requested. The advantage of the CCFM.VENTS method is that much less storage is required to archive scenario calculations. CFAST uses the former since many other programs depend upon the history file format. An alternate strategy is to compress binary history files. It has been found that these files can be compressed from 20 to 95 percent.

solver The solvers used in CCFM.VENTS and FAST are different. A potential exists for an improvement in the performance of CFAST by incorporating some of the solution techniques used by CCFM.VENTS. These techniques will be tested in a subsequent project.

global error checking Any difference in a conserved quantity such as the total mass from one time step to the next represents a good estimate of the *global error*. In general, solvers for ordinary differential equations can only estimate and therefore reduce *local error*, the error incurred since the last time step. This global error estimate is a good check on the solver, and physical calculation routines. CCFM.VENTS calculates the total mass in all rooms

of a simulation each time results are printed out. An equivalent capability exists in FAST, although the energy is the quantity which is tracked. These are good checks on the equations that are being solved. Once a model is operational, the calculation is extraneous. As such it is not currently implemented in CFAST. However, since the purpose of making the CFAST generally available is to allow researchers to change the internal structure such as the basic equations that are being solved, this feature will be re-incorporated into CFAST.

data allocation The methods used by FAST and CCFM.VENTS to store program variables differ. CCFM.VENTS allocates memory dynamically while FAST allocates it statically. **Dynamic memory allocation** means that the memory used by a subroutine's variables is allocated only when it is needed usually upon entering the subroutine. Memory is usually released when control is returned to the calling routine. On the other hand, when using **static memory allocation**, memory is reserved at the beginning of program execution and is available throughout the entire run. The main effects of these two strategies are as follows. First, the amount of memory necessary to run a static program is larger than a dynamic one. Second, local subroutine variables[4] in a dynamic program lose their values across subroutine calls.

To convert a program from one that uses static to one that uses dynamic data allocation, one must track down those subroutine variables whose values must be preserved across subroutine calls. These variables are typically counters, variables that are incremented each time a subroutine is called. If necessary, subroutine variables can be saved across subroutine calls by putting them in FORTRAN SAVE statements. CFAST uses dynamic memory allocation. There was a strong incentive to use this method since the size of the executable was reduced by about 10%. The caveat to keep in mind is that routines should not make the assumption that local data will be preserved. If local data must be preserved, then the PARAMS or CENVIRO common block should be modified to keep the data. These will be saved. Judicious use must be made of this facility, however, for CFR reserves the right to incorporate routines at its discretion, and routines that clutter up the structure are not likely candidates for inclusion.

1.2 Overview

Section 2 discusses the structure of CFAST and the environment used to develop it. Section 3 discusses how to make changes in an orderly fashion. The control program is listed with annotations, since this is crucial for anyone who wants to add (or delete) physical routines. Section 4 presents material on protocols used by the CFAST suite of programs. Section 5 documents the input process used by the routine NPUTP.

4 A subroutine variable is local if it only occurs inside the subroutine. Some examples of subroutine variables that are not local are variables that occur in common blocks and variables that are passed through the subroutine's argument list.

The appendices document various internal aspects of CFAST. Appendix A documents the parameters read by the subroutine NPUTQ. Appendix B documents the global data structures contained in the labeled common blocks. Appendix C shows how subroutines are related. (This appendix was generated from the tool, **roadmap**.) Finally, Appendix D documents the purpose of each routine. It forms a subroutine glossary.

2. STRUCTURE OF CFAST

There are four aspects of the structure of CFAST that will be discussed in this section. These aspects are the subroutine, the program suite, the directory and the network structure. These structural aspects are of interest to the person wishing to modify and to enhance the model. The **subroutine structure** is the relationship of the various subroutines. That is, what subroutines call what subroutines. The **program suite** is the relationship of the program modules CF_in, CF_set, CFAST, CF_rpt, CF_plot, and CSHELL to one another. The **directory structure** delineates the relationship amoung the subrountines and main programs. It deals with the location of subroutines in which directories, based on which main modules them. The **network structure** is a statement of the location of files that are a part of CFAST. Some files are located on the file server[5] and others are located on a local computer. By keeping a common copy of a program on the file server, required changes to update the program need only be made in one place. Keeping a program on several computers can result in confusion as to who has the correct version. Further, our structure, and the way the makefiles are structured, allows us to run the whole process on the server, on an individual workstation, or in a mixed environment. The whole process is transparent to the developer.

The structures set up in this section will be used in section 3 to show how to make changes in an orderly manner. The techniques used here to develop CFAST can be applied to other software development projects. The extra steps taken are especially important for programming projects that involve several people over an extended period of time.

2.1 Subroutine Structure

The calling sequence for a suite of procedures defines their relationship to the calling program and to order in which they are called. A detailed map of this information for the model program, CFAST, is given in Appendix C. It was felt that the structural details of the model portion of CFAST were most important since this is the program in the CFAST suite of programs that is most likely to be modified. The subroutine structure of CFAST is illustrated in figure 1. It was derived from Appendix C which was produced using **roadmap** [3]. A document analogous to Appendix C can be produced for any other program in the CFAST suite by using **roadmap.**

The model can be split into distinct parts. There are routines for reading data, calculating results and reporting the results to a file or printer. The major routines for performing these functions are identified in figure 1. The algorithms for those routines, identified in figure 1 as physical routines or auxiliary physical routines, are documented in reference [2]. These routines calculate quantities such as mass or energy flow at one

5 A file server is a computer that shares disk storage with other computers, usually over some type of network.

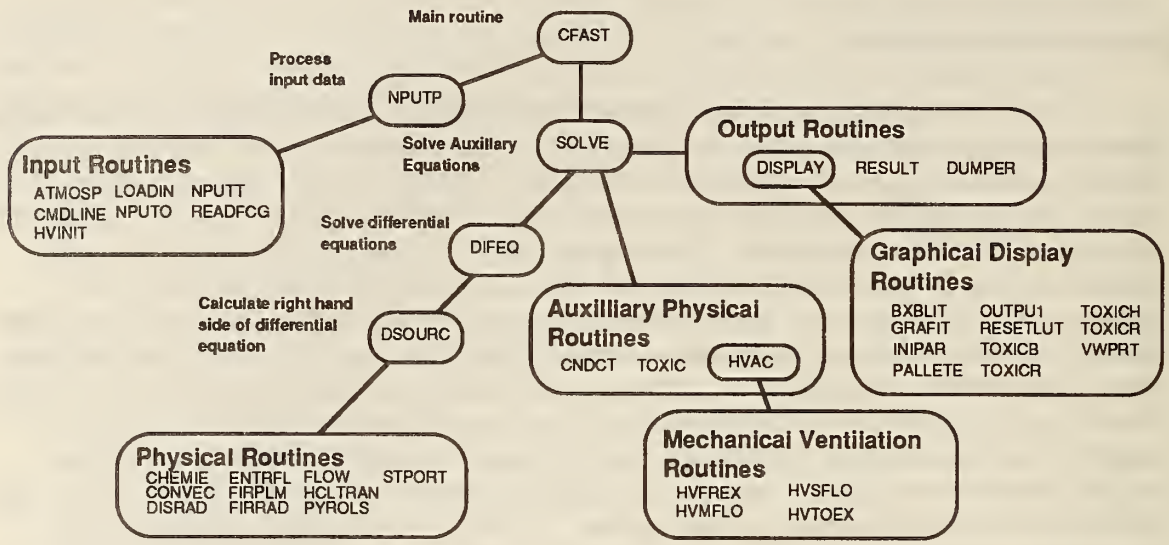

Figure 1. Subroutine Structure and Interaction.

particular point in time for a given environment.

The routines SOLVE, DIFEQ and DSOURC are the key to understanding how the physical equations are solved. SOLVE is the control program that oversees the general solution of the problem. It invokes DIFEQ which calls DSOURC to solve the transport equations. Then SOLVE does the mechanical ventilation (HVAC), conduction (CNDUCT0 and toxic dose (TOXIC) calculation. An outline of this solution process is illustrated in figure 2. The problem that these routines solve is as follows. Given a solution at time (t) what is the solution at time (t + Δt)? The transport equations are differential equations of the form

$$\frac{dy}{dt} = f(y,t) \qquad (1)$$
$$y(t_0) = y_0$$

where y is a vector function representing pressure, layer height, mass, etc. and f is a vector function that represents changes in these values with respect to time. The term y_0 is an initial condition at the initial time t_0. CFAST uses the subroutine DSOURC to compute the right hand side of eq (1). DSOURC is discussed in section 3.5. The algorithm for advancing the solution to the fire problem then is to solve the ODE for some time interval (t,t+Δt). This computed solution is then used to solve the heat conduction and mechanical ventilation problem over the same time interval.

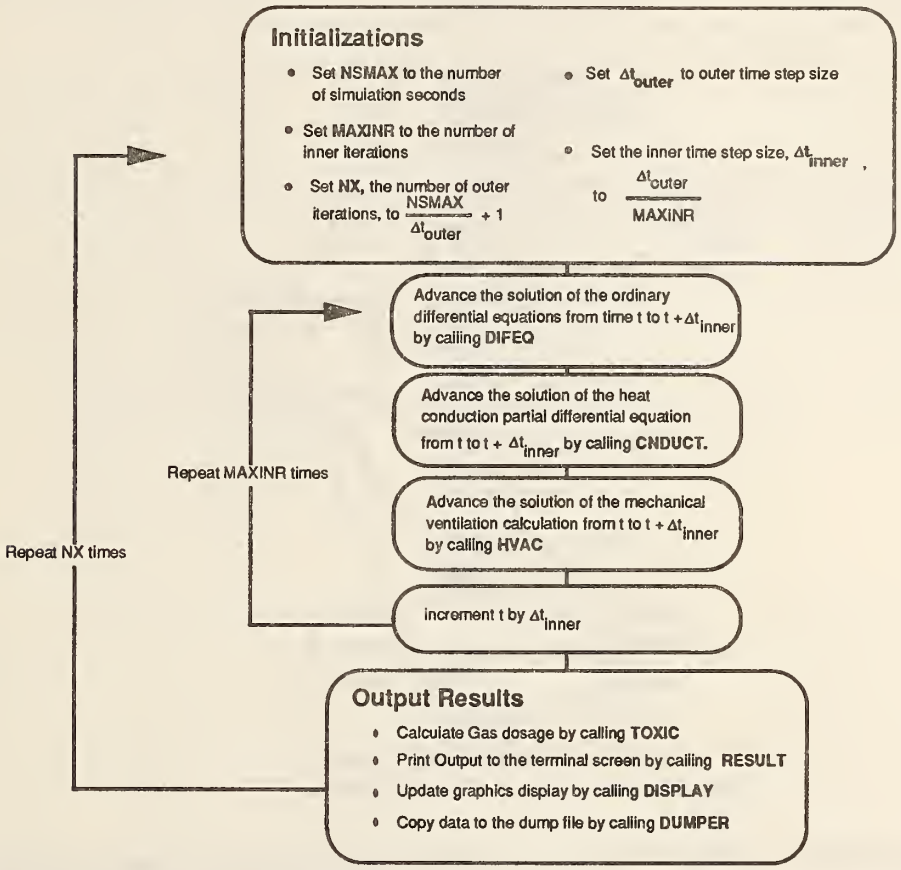

Figure 2. Routine SOLVE Computations.

Note that there are several distinct time scales that are involved in the solution of this type of problem. The fastest will be chemical kinetics. We avoid that scale by assuming that the chemistry is infinitely fast. The next larger time step is that associated with the flow field. These are the equations which are cast into the form of ordinary differential equations. Then there is the time scale for mechanical ventilation, and finally, heat conduction through objects. This technique of splitting the solution into parts associated with a particular time scale is called "time splitting." It is well established, and works quite well so long as one bears in mind the fact that the time step must be smaller than the time scale associated with the "next" phenomena. By way of example, chemical kinetics are typically on the order of milliseconds. By choosing a time step of 0.1 s, we have satisfied the criterion that the chemistry is infinitely fast. The transport time scale will be on this order. The mechanical ventilation and conduction time scales are typically several seconds, or even longer.

2.2 Program Suite Structure

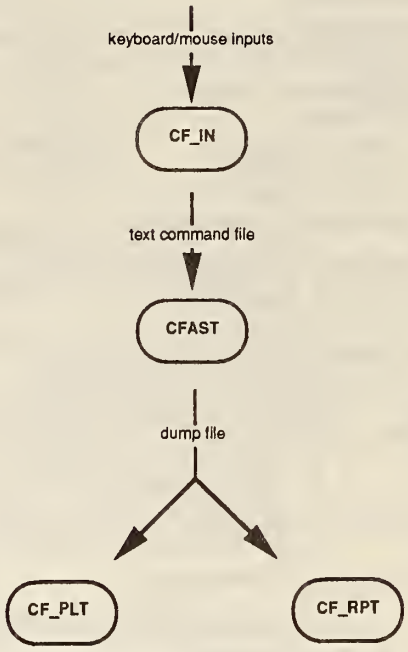

Figure 3. Overall Program Structure.

The CFAST suite consists of several programs. Each one serves a specific function. CF_IN is a program that solicits input from the user in a friendly manner using full screen windowing techniques. This program creates a text file readable by CFAST. CFAST takes a fire scenario created with CF_IN and predicts the environmental conditions over a period of time. CFAST then stores the results in a history file that can be read by CF_PLT and CF_RPT. CF_PLT generates plots of data that were previously generated by CFAST. These routines can be run independently, or together though the CSHELL program. The relationship between the programs in CFAST is illustrated in figure 3.

The current version of these programs run on a MSDOS compatible micro-computer. It is planned to have CFAST run on other platforms such as the Macintosh or a UNIX work-station. The planning for this "porting" process is underway.

2.3 Directory Structure

The model is organized by functional element. There is a directory entry for each main module, as well as entries for common routines and utility functions. The present

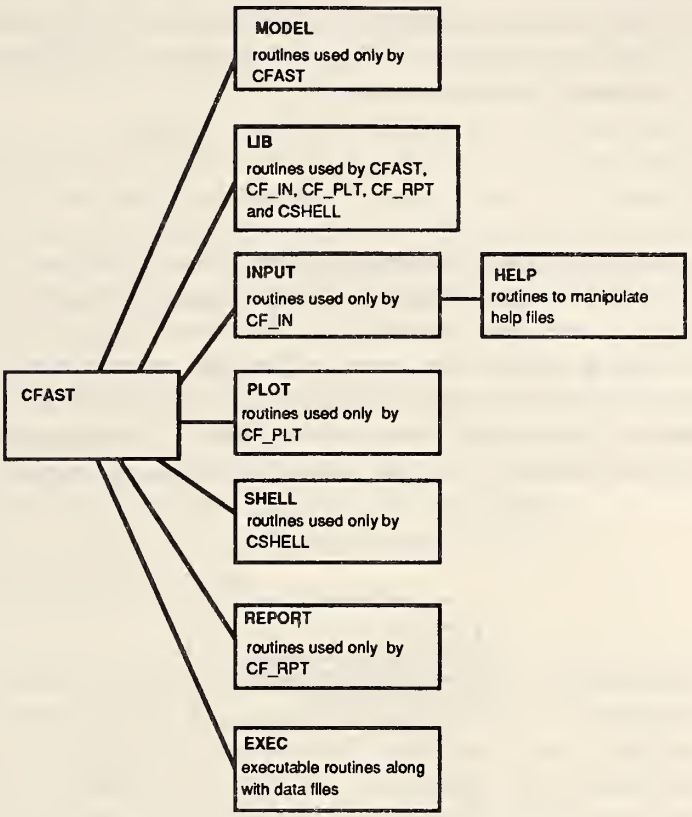

Figure 4. Basic Directory Structure of CFAST.

directory structure is illustrated in figure 4.

As it appears, this structure can occur on a single workstation, or on a single work-station connected to a network. In the latter case a parallel structure would be on the network and workstation and appropriate routines would be accessed from either place. This mode of operation is transparent to the user.

Using the tool **fdiff**, many routines in CF_in, CF_plot and CFAST were identified as being essentially identical. Routines that were identical were moved to a directory called LIB (for library). These routines are accessible to other programs in the CFAST suite. This allows required changes to be made in only one place, but even more importantly, only once. If two routines had the same name but were different then the names of one or both the routines were changed. This was done to avoid the confusion of having two routines with the same name but serving different functions.

The directory CFAST\INPUT contains the routines used only by CF_in. The directory CFAST\PLOT contains the routines used only by CF_plot. The directory CFAST\MODEL

contains the routines used only by CFAST. Any routines that are used by two or more programs in the CFAST suite are contained in the directory CFAST\LIB. The directory CFAST\SHELL contains the routines for the shell program, CSHELL, and auxiliary programs for accessing and displaying the data structures.

2.4 Network Access

The basic strategy for storing files is to keep a master copy of CFAST on a network file server. Any person wishing to modify a CFAST routine gets a base copy from the network and modifies it on a local workstation. *The master copy of CFAST remains unchanged.* Only a single developer has access to this local version, which may then be modified. Such an implementation allows several people to work on CFAST without interference. Typically, the developer may modify only a few routines. The next section discusses techniques that take advantage of this fact when building a new version of CFAST.

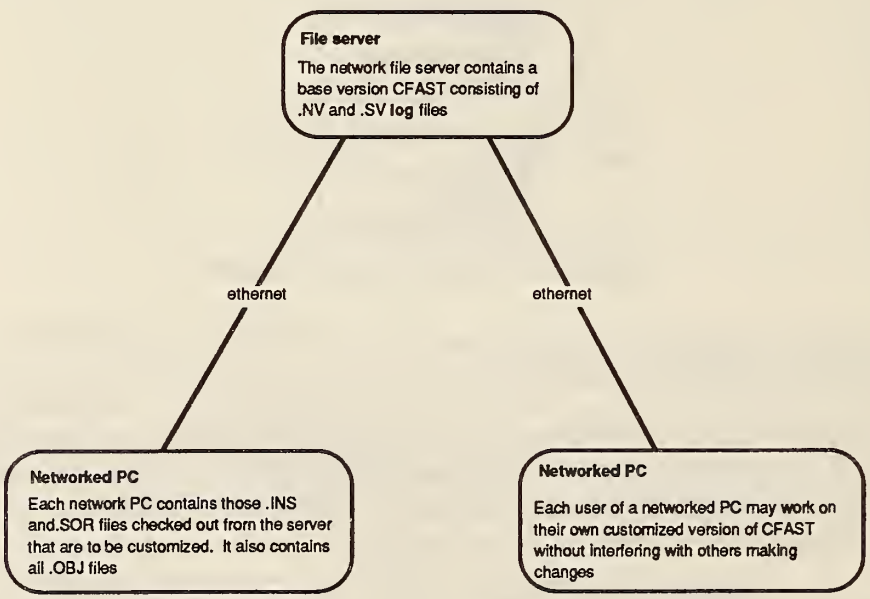

Figure 5. Network Usage.

The strategy for storing CFAST files on the network is illustrated in figure 5.

3. UPDATING THE MODEL

In general there are suggestions throughout this text on procedures to follow, and techniques that should be used, and so on. For example, we discuss later how to modify (and also what not to do to) the configuration file. In order to make the model more portable, and easier to modify and test, we have incorporated a set of guidelines that must be followed. The guidelines were developed so that several lines of development could take place simultaneously, and still be integrated as a final product.

The points considered here are:

Naming conventions,
Data and parameter passing,
Variable and procedure (subroutine & function) names,
Physical routines, where they go and how to modify them,
Precision of data (internal), and
Multiple platforms.

naming conventions In general, variable and procedure names should reflect the substance of the variable, or the use for a procedure. Given the limitation in names prescribed by the ANSI FORTRAN standard [4], one has to use a judicious choice to reflect the content of a variable or procedure. Indeed, the ANSI standard limits names to six characters. As most compilers allow eight or more characters, we have used the long names, although somewhat sparingly. However, in all cases, the names are unique within the first six characters. Thus when moving the model to a compiler which strictly enforces the standard, the model will still run correctly if variable names are truncated to six characters.

The environment common blocks contained in the include file CENVIRO.INC, can be differentiated from the model data, by the stylized form of the variable names. All variables begin with ZZ and are followed by a four character name, once again indicative of the meaning of the variable.

data and parameter passing In general routines should not write to common block variables. It is particularly important to observe this rule for the solver data structure, since it can change without notice. The most common data that physical routines (PIR) will need can be obtained from the environment common block, which is used in a *read only* mode. Otherwise, data should be passed into the routine through the procedure header (calling sequence or argument list), and results returned through the header. Some data, particularly static data such a the total volume of a compartment, is not in the environment data. In these cases the appropriate data can be obtained from the primary common blocks. The one routine that normally writes to the common blocks is DSOURC, the control program. Thus any model data that needs to be changed should be called here, and the data structures

updated here. Following this convention will at least minimize the maintenance headache of tracking errant data assignments.

variable and procedure (subroutine & function) names In most cases we have used names which are six or fewer characters. This is in conformance with the ANSI standard for FORTRAN. As mentioned above, there are a few cases where this would render the purpose of the variable or procedure completely obscure. In those cases, we extended the name, never exceeding eight characters. In all cases, though, the names are unique to the first six characters. This rule must be followed!

physical routines, where they go and how to modify them A physical routine is one that calculates a source term for the solver. In reference [2], these are eqs (1-4). **All** of these routines are called through the control program DSOURC. In almost all cases, the data needed and returned by the routine is passed through the procedure header. There is some effort to make the code readable, so some of the information needed by the routine is obtained directly by routine from the envrionment common blocks for time dependent data, or the primary common block for static data.

Routines which incorporate time splitting are referenced in the main solver control program (SOLVE). At present these latter constitute a very small set, namely DIFEQ (the link to DSOURC), the mechanical ventilation (HVAC), the species report (TOXIC), and the heat conduction routine (CNDUCT).

Any physical routine can be processed through DSOURC. Section 3.4 discusses this routine. There are three sections. The first two calculate the actual physical source terms, and the third sorts these forcing terms into the form necessary for the solver. The first section is the fire itself, and the second section is everything else. This latter is a loop over compartments, whereas the former is a loop over fire sources. If the PIR is associated with the fire, then the first section is most appropriate, otherwise, the second section should be utilized. There are two mechanisms for adding terms. The simplest is to calculate a local variable, and then add it to the appropriate place in section three which sorts the terms for the solver. Alternatively, one of the existing variables can be utilized. This is the preferred way, but care must be taken in placing the initialization of such variables in appropriate places. It won't do much good to add a source term if the variable is subsequently initialized to some specific value, such as zero.

precision of data (internal) It was decided that is should be simple to switch between single and double precision. This statement applies to variable names as well as function names. The rationale is that although the normal operation will be single precision, one might find that double precision is necessary for new algorithm development, or debugging purposes. The conversion will be easy. The rules to achieve this are

1. every subroutine should have an IMPLICIT statement at the beginning; IMPLICIT REAL (A-H,O-Z) for the real version of CFAST and IMPLICIT DOUBLE PRECISION (A-H,O-Z) for the double precision version.
2. Do not pass floating point constants to subroutines. Instead set a variable equal to the constant and then pass that variable
3. All floating point constants used in the program should be of type double precision, e.g. 1.0D0 not 1.0
4. For floating point variables do not over-ride the default type, i.e. variable names beginning with I, J, ... N should always be of type integer.
5. When using intrinsic functions (SIN, COS) always use the generic form (not DSIN, DCOS, DABS, etc.).

If the above guidelines are followed then to switch precision one simply changes every statement of the form

```
IMPLICIT REAL (A-H,O-Z)
```

to

```
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

or *visa versa*. Implicit statements are contained in the include file, PRECIS.INC. The filter file version, PRECIS.INS, is given by

```
%IF DOUBLE
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
%ELSE
      IMPLICIT REAL (A-H,O-Z)
%ENDIF
```

There are two key words associated with this guideline that are used by the filter process, that is in the translation from .SOR files to .FOR files. They are DOUBLE and SINGLE. Actually, SINGLE is simply the absence of DOUBLE, so is implemented as the ELSE clause shown above. If the DOUBLE key is set when invoking FILT then PRECIS.INC will contain the IMPLICIT DOUBLE PRECISION statement. Otherwise PRECIS.INC will contain the IMPLICIT REAL statement. All routines that use any model data must follow this convention. The single variable we found easiest to overlook is the XI in the CFIO common block. It is the only **real** variable in that data structure. The way to check whether there is access to the model data structures is to use COMCHECK [3].

multiple platforms There are three key words defined at present: IBMPC, MAC and UNIX. Any code which might be sensitive to the computer platform should use such dependent clauses. As other computers are utilized, this list will be expanded. Platform dependent code is present in the routines TOUPPER and TOLOWER and in the CPARAMS.INC include file. CPARAMS.INC contains a CHARACTER*1 variable, FILSEP, that is used to separate names within a full path name. FILSEP is '\' for the IBMPC, ':' for

the Macintosh and '/' for UNIX machines. The graphics routines and code to open the "terminal screen" for output are likely candidates for this type of enchancement in the future.

3.1 Tools and File Types

It was desirable in developing this model to avoid making software changes in a haphazard manner. For example, it is easy to lose track of what changes were made to a large file (say bigger than 5000 lines) when editing it directly. It is then hard to un-do these changes if it is later discovered that they were incorrect and it is also difficult for more than one person to work on a program at the same time. This section indicates how some of these problems can be eliminated. The file types and tools used to convert between these file types are illustrated in figure 6.

In order to maintain an orderly updating procedure, a revision control system is used. The one described here is called the Polytron Version Control System, PVCS [5], but the ideas are generally applicable to other such version control systems. The base version of each routine is kept in a log file. The **log file** contains the current version, and deltas or change images to get back to previous versions. In addition, there are comments which indicate the reason for the changes. Also contained in the log file is an audit trail of who made changes and when they were made. Such a facility allows one to revert back to an earlier version if some side effect of a change produces incorrect results.

Log files, current versions and compiled code are kept together. If a network is being used, then the log files will be on the network, but the current version under revision, as well as the compiled code, will be in the current directory on the current workstation.

To facilitate an orderly update process, several tools that are described in this section are used to convert from one file type to another. The file types that will be discussed are log files, filter files, FORTRAN source files, object files and executable files. Software tools are also described that convert from one file type to another. These tools are **get, put, filt** [5], **f77l** [6] and **plink86** [7] Some of these tools and file types will be familiar to anyone who has developed FORTRAN programs. Others were used in the CFAST project to allow easier development in a multi-developer environment.

In order to understand the tools, we must first understand the nuance of the types of files that are used. A **log file** contains a record of all changes made to a routine. The developer "checks out" a log file to make changes. After being satisfied that the changes are correct, the module is "checked in" as a new version. The original and the new version are both contained in the log file for that routine. Someone else can then either check out the original or revised version. Log files for these routines have the extension .SV while log files for include files have the extension .NV. Note that log files are not edited in the usual way that one edits a FORTRAN program. Only the files that are checked out with the **get** tool can be modified.

A log file is checked out using the tool **get**. Changes are checked back in using the tool **put**. Both of these tools are described in the PCVS manual [5]. Again, similar tools exist on other platforms. When a log file is checked out using the **get** command it is converted to a filter file. This will be either an .SOR or .INS file. The files maintained in the log file system contain editing codes. That is, files checked out of the .SV and .NV **log** files are .SOR and .INS files and it is these latter that are actually maintained and edited. However, they need to be converted prior to actual use. The **filt** tool provides this service. The makefiles which are discussed below handle it transparently. The files become .FOR and .INC files respectively. The .FOR and .INC files should not be edited directly. They are not maintained, nor saved.

An summary of several modifications of the module PYROLS looks like that shown in figure 7.

To check out a log file for the routine HVVIS contained in the program CFAST use the command

get -L CFAST\LIB\HVVIS.SV

to obtain the filter file, HVVIS.SOR. The '-L' option allows the checked out file to be modified. HVVIS is used by both CF_IN and CFAST, so it is located in the LIB directory. To preserve the structure, it should be checked into the local CFAST\LIB directory. The header of the *log* files looks like

```
Logfile:          E:\LIB\hvvis.sv
Workfile:         hvvis.sor
Owner:            wwj
Last trunk rev:   1.2
Locks:            wwj : 1.2
Rev count:        3
```

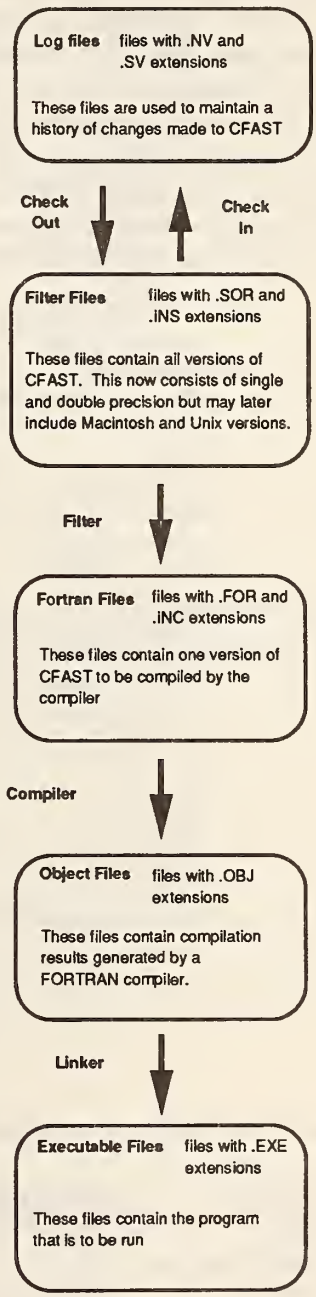

Figure 6. File Types Used To Maintain CFAST

```
Logfile:          E:\LIB\pyrols.sv
Workfile:         pyrols.sor
Owner:            wwj
Last trunk rev:   1.2
Locks:
Rev count:        3
Attributes:
   WRITEPROTECT
   CHECKLOCK
   NOEXCLUSIVELOCK
   NOEXPANDKEYWORDS
   TRANSLATE
   COMPRESSDELTA
   COMPRESSWORKIMAGE
Version labels:
Description:
for->sor for filtering; from 18.5
----------------------------------
Rev 1.2
Checked in:     28 Mar 1990 08:55:22
Last modified:  13 Mar 1990 09:54:42
Author id: wwj     lines deleted/added/moved: 39/41/0
change the calling procedure to pass new species production parameters
hcrtt, oxygen, ...
----------------------------------
Rev 1.1
Checked in:     28 Mar 1990 08:53:46
Last modified:  07 Mar 1990 16:57:22
Author id: gpf     lines deleted/added/moved: 29/39/0
1.  convert to double precision
2.  remove common block writes from source routines
    (routines called by DSOURC)
----------------------------------
Rev 1.0
Checked in:     28 Mar 1990 08:51:04
Last modified:  14 Dec 1989 16:45:36
Author id: wwj     lines deleted/added/moved: 0/0/0
Initial revision.
==================================
```

Figure 7. Revision control history.

and what it looks like if two developers are working at the same time

```
Logfile:          E:\LIB\hvvis.sv
Workfile:         hvvis.sor
Owner:            wwj
Last trunk rev:   1.2
Locks:            wwj : 1.2
                  gpf : 1.1
Rev count:        3
```

There are three types of changes that one might expect to find in a software system that is to be used on more than one platform. One is the global difference over the entire model that can occur because of some decision in the way the model is to be operated. An

example of this is using a model in single or double precision. Another difference is that needed to port the model to another platform. A third would be change to include enhancements, or to fix bugs. The latter would be incorporated by specific changes in the modules. The second would be through different versions of the same program or library tailored to run on different computer. The first would be by conditional clauses in the source. Some of these are implemented by a technique known as filtering. Slight changes in the software are usually required when a program is run on a different computer. For example, the FORTRAN OPEN statement and floating point characteristics may be different in different versions. The filter file can contain machine dependent or independent code within "IF" statements of the form

```
%IF DOUBLE
        .
        .
        .
   double precision dependent code
        .
        .
        .
%ENDIF
```

The string, 'DOUBLE' is an example of a key. If this key is set when executing the filter program then text between the %IF DOUBLE and %ENDIF statements will be copied to the output files. The filter file for the routine is HVVIS is given below.

```
%IF DOUBLE
      DOUBLE PRECISION FUNCTION HVVIS(T)
%ELSE
      FUNCTION HVVIS(T)
%ENDIF
      INCLUDE 'PRECIS.INC'
C
C     FUNCTION CALCULATES ABSOLUTE VISCOSITY (PASCAL SECOND)
C     BY INTERPOLATION FOR 200 TO 2000 DEG K.
C     DATA FROM NASA TECH NOTE D-7488 BY POFERL, D. J. AND
C     SVEHLA, R. ,1974 EXCEPT FOR THE VALUE AT 200 DEG K
C     WHICH IS FROM THE CRC HANDBOOK OF CHEMISTR & PHYSICS,
C      66TH ED.,1985 PAGE F-42.
C
C     T=ABSOLUTE TEMPERATURE IN DEGREES KELVIN
C
      LOGICAL ERROR
      DIMENSION V(19)
%IF DOUBLE
      DATA V/13.1D0,18.4D0,22.7D0,26.5D0,29.9D0,33.1D0,36.2D0,39.1D0,
     +       41.9D0,44.5D0,47.0D0,49.4D0,51.7D0,54.0D0,56.3D0,58.5D0,
     +       60.7D0,62.9D0,65.1D0/
%ELSE
      DATA V/13.1,18.4,22.7,26.5,29.9,33.1,36.2,39.1,41.9
     +,44.5,47.0,49.4,51.7,54.0,56.3,58.5,60.7,62.9,65.1/
%ENDIF
      DATA ERROR/.TRUE./
C
      I=T/100.D0 - 1.D0
      IF(I .GE. 19)GO TO 1
      IF (I.LT.1) THEN
         IF (ERROR) THEN
```

```
            WRITE(*,2) T
2           FORMAT (' OUT OF VISCOSITY INTERPOLATION RANGE = ',F10.3)
            ERROR = .FALSE.
          ENDIF
          I = 1
        ENDIF
        T1 = 100.D0*(1.D0+FLOAT(I))
        IM = I + 1
        HVVIS = (V(I) + (V(IM) - V(I))*(T - T1)*0.01D0)*1.0D-6
        RETURN
1       HVVIS=V(19)*1D-6
        RETURN
        END
```

Using filters makes it easier to maintain multiple versions of a program within one file. Filter files for CFAST routines have an extension of .SOR while filter files for include files have a .INS extension. The filter keys presently used in CFAST are DOUBLE and IBMPC. It is anticipated that other keys such as UNIX and MAC will be used in future versions of CFAST. These keys will denote sections of the programs that are specific to a UNIX workstation and the Macintosh.

The key, DOUBLE, allows CFAST to be switched between single and double precision. FILT is the name of the program used to filter CFAST routines. FILT produces FORTRAN routines with .FOR or .INC extensions. To obtain the double precision version of HVVIS use the command

FILT -KDOUBLE,IBMPC HVVIS.SOR HVVIS.FOR

The string '-K' followed by a list of keys (in this case DOUBLE and IBMPC) specifies the keys to be set. This command copies the text contained in the filter file, HVVIS.SOR, to the FORTRAN source file, HVVIS.FOR. This file is shown below. Notice that it looks similar to HVVIS.SOR except for the statements within the '%IF' blocks. FORTRAN statements between the %IF DOUBLE and %ELSE statements are copied to the output file while statements between %ELSE and %ENDIF are not.

```
      DOUBLE PRECISION FUNCTION HVVIS(T)
      INCLUDE 'PRECIS.INC'
C
C     FUNCTION CALCULATES ABSOLUTE VISCOSITY (PASCAL SECOND)
C     BY INTERPOLATION FOR 200 TO 2000 DEG K.
C     DATA FROM NASA TECH NOTE D-7488 BY POFERL, D. J. AND
C     SVEHLA, R. ,1974 EXCEPT FOR THE VALUE AT 200 DEG K
C     WHICH IS FROM THE CRC HANDBOOK OF CHEMISTR & PHYSICS,
C      66TH ED.,1985 PAGE F-42.
C
C     T=ABSOLUTE TEMPERATURE IN DEGREES KELVIN
C
      LOGICAL ERROR
      DIMENSION V(19)
      DATA V/13.1D0,18.4D0,22.7D0,26.5D0,29.9D0,33.1D0,36.2D0,39.1D0,
     +       41.9D0,44.5D0,47.0D0,49.4D0,51.7D0,54.0D0,56.3D0,58.5D0,
     +       60.7D0,62.9D0,65.1D0/
```

```
      DATA ERROR/.TRUE./
C
      I=T/100.D0 - 1.D0
      IF(I .GE. 19)GO TO 1
      IF (I.LT.1) THEN
         IF (ERROR) THEN
            WRITE(*,2) T
2           FORMAT (' OUT OF VISCOSITY INTERPOLATION RANGE = ',F10.3)
            ERROR = .FALSE.
         ENDIF
         I = 1
      ENDIF
      T1 = 100.D0*(1.D0+FLOAT(I))
      IM = I + 1
      HVVIS = (V(I) + (V(IM) - V(I))*(T - T1)*0.01D0)*1.0D-6
      RETURN
1     HVVIS=V(19)*1D-6
      RETURN
      END
```

The FORTRAN compiler translates files with .FOR and .INC extensions to object files with .OBJ extensions. The .OBJ files are converted to an .EXE file using a linker. The last two steps are the same in any programming project. It is important to understand why the first two steps, *i.e.*, using **get** and **filt**, were taken in the CFAST development cycle. First, log files allow one to keep track of changes made to a routine over a period of time. They allow one to "back up" to earlier versions of the software. They allow changes made by several persons to be merged into one version less painfully. This has already been done once. The two authors of this report each made changes to CFAST. These changes were later merged using the tool, VMRG. This tool automatically identified where we both made change to the same line of code. This is called a collision. Only the "collisions" had to be merged manually. The process of merging corrections is discussed in section 3.3.

Second, using filter files makes it easy to keep multiple versions of a program in one file. This eases the maintenance burden of updating portions of a program that are common to all versions, since the same changes do not have to be made in several places.

3.2 Rebuilding CFAST

A program has to be rebuilt after its source code has changed. The traditional method for regenerating a program is to keep the entire program in one file. This would be over 9,000 lines of code in the case of CFAST. A disadvantage of keeping the whole program in one file is that the compiler would process every subroutine even though only a few may have changed. This is needlessly inefficient since the edit-compile-link-run cycle would be much longer than necessary. This cycle should be shortened as much as possible in order to improve programmer productivity . Two ways of doing this are to use faster computers and to minimize unnecessary work by only compiling routines that have changed. The rest of this section will concentrate on methods for accomplishing the latter.

To rebuild CFAST use the command

MAKE -F CFAST

The programming tool, MAKE [8], is used to rebuild (make) programs. It uses a set of rules contained in a special file called a *makefile* to determine how to rebuild files that are out of date. MAKE determines which files are out of date by examining modification times of various files. These files are specified in the *makefiles*, in this case CFAST.MAK.

An example will help clarify the terminology. Suppose that HVVIS.OBJ depends on HVVIS.SOR. That is, whenever HVVIS.SOR changes HVVIS.OBJ needs to be regenerated. This dependency relationship could be specified in a *makefile* as

```
HVVIS.OBJ : HVVIS.SOR
        FILT HVVIS.SOR HVVIS.FOR
        F77L HVVIS.OBJ
```

The **filt** and **f77l** commands are executed whenever HVVIS.SOR has been changed more recently than HVVIS.OBJ. This is an example of a specific rule. It only applies to one particular file. General rules are used to specify how to make files of one extension out of another. For example, the following general rule could be used to make a .obj file out of a .for file

```
.for.obj :
        f77l $<
```

The character string '$<' is a macro used to specify the name of the routine to be compiled. A list of general rules is contained in the file BUILTINS.MAK. Some of these rules are given by

```
# BUILTINS.mak


# This rule tells how to make a .SOR file from a .SV file
# using POLYTRON PVCS with .sv files.
#
.sv.sor :
        =get -q -w $<($@)

# This rule says that a .FOR file should be created from
# a .SOR file by using FILT

.sor.for :
         filt -kIBMPC,DOUBLE $*.sor $*.for

# This rule says a .OBJ file should be created by
# using the Lahey FORTRAN Compiler on a .FOR file.

.for.obj :
          f77l $<
```

```
.nv.ins :
        =get -q -w $<($@)

.ins.inc :
         =filt -kIBMPC,DOUBLE $*.ins $*.inc

# END
```

Two rules in the above listing of BUILTINS.MAK invoke the tool **filt**. These tools are configured to create double precision versions of CFAST routines. To create single precision version remove the key, DOUBLE, by changing the respective commands

```
=filt -kIBMPC,DOUBLE $*.sor $*.for
=filt -kIBMPC,DOUBLE $*.ins $*.inc
```

to

```
=filt -kIBMPC $*.sor $*.for
=filt -kIBMPC $*.ins $*.inc
```

The file CFAST.MAK contains a list of rules that specify which object files are candidates for rebuilding. It also specifies how to build the program CFAST from these object files by using the linker, **plink**. The file CFAST.MAK is given by

```
.NOSHELL
.NOREMAKE
.LOGFILE .SV(.SOR -R)
.LOGFILE .NV(.INS -R)
.PATH.SV =  .;..\LIB
.PATH.NV =    ..\LIB
.PATH.SOR = .;..\LIB
.PATH.INS =   ..\LIB

ob1 =cfast.obj initdi.obj
ob1a =$[s,",",$(ob1)]
obe =solve.obj lenoco.obj display.obj dsourc.obj difeq.obj toxic.obj
obea =$[s,",",$(obe)]
ob2 =disrad.obj toxicb.obj toxich.obj toxicr.obj ttcb.obj vwprt.obj wddraw.obj
ob2a =$[s,",",$(ob2)]
ob3 =dumper.obj entrfl.obj firplm.obj firrad.obj flow.obj writeot.obj
ob3a =$[s,",",$(ob3)]
ob4 =flwout.obj frflow.obj grafit.obj hsets.obj outpu1.obj
ob4a =$[s,",",$(ob4)]
ob5 =palette.obj polnum.obj pyrols.obj result.obj stport.obj
ob5a =$[s,",",$(ob5)]
ob6 =getview.obj bxblit.obj chemie.obj cnduct.obj convec.obj
ob6a =$[s,",",$(ob6)]
ob7 =loadin.obj loadup.obj nputo.obj nputp.obj nputq.obj disclaim.obj toupper.obj
ob7a =$[s,",",$(ob7)]
ob8 =nputt.obj atmosp.obj convrt.obj datype.obj readop.obj readcf.obj openshel.obj
ob8a =$[s,",",$(ob8)]
ob9 =readas.obj readcv.obj readin.obj restrt.obj dreadin.obj sstrng.obj
ob9a =$[s,",",$(ob9)]
oba =disthe.obj gasload.obj diferr.obj inipar.obj setlut.obj f_info.obj
obaa =$[s,",",$(oba)]
```

```
obb =resetlut.obj cmdline.obj readfcg.obj hvinit.obj mvout.obj bstrng.obj grquery.obj
obba =$[s,",",$(obb)]
obc =hvac.obj hvmflo.obj hvtoex.obj hvfrex.obj hvsflo.obj hvvis.obj
obca =$[s,",",$(obc)]
obd =hvfric.obj hvfan.obj hcltran.obj
obda =$[s,",",$(obd)]
obinc =cfast.inc display.inc params.inc precis.inc cparams.inc cfio.inc cshell.inc thermp.inc
obins =$[f,"",$(obinc),"ins"]

..\EXEC\cfast.exe: $(obinc) $(ob1) $(obe) $(ob2) $(ob3) $(ob4) $(ob5) $(ob6) $(ob7) $(ob8)
$(ob9) $(oba) $(obb) $(obc) $(obd)
 PLINK <@<
file $(ob1a),dvblok.obj
 allocate device,f77l,metddhfl,wndodgr
 library  device,f77l,metddhfl,wndodgr
begin     section preload file $(obea)
   begin    file $(ob2a)
            file $(ob3a)
            file $(ob4a)
            file $(ob5a)
            file $(ob6a)
            file $(obca)
            file $(obda)
          section preload file $(ob7a)
    file $(ob8a)
    file $(ob9a)
    file $(obaa)
    file $(obba)
   end
end
<
 +mv cfast.exe ..\exec

$(ob1) $(obe) $(ob2) $(ob3) $(ob4) $(ob5) $(ob6) $(ob7) $(ob8) ...         $(obd) : $(obinc)

$(obinc) : $(obins)
```

These make files were designed to cause **make** to search the local workstation first for files in CFAST. If the necessary files were not found then **make** searches the network file server. This allows customized versions of CFAST to be maintained on the local workstation and commonly used routines to be used on the file server.

3.3 Merging Corrections

After two or more people have made changes and are satisfied that the changes are correct it is necessary to merge these changes together into one working version. There are two tools available to assist in this process. These tools are **vdiff** and **vmrg**. **vdiff** examines two files reporting any differences in the files. **vmrg** takes two modified files along with the base from which they were modified and produces one file that contain both sets of corrections. Several things can happen to complicate the merger process. First, two people may modify the same line of code. Second, one person may change a line of code that the other person is depending on. The second problem is the more difficult to deal with since it cannot be detected automatically. A solution to this type of problem depends upon effective communication between persons making changes to the software.

The first problem, two people changing the same line of code, can be detected automatically by **vmrg**. It does this by highlighting the section of code modified by both persons.

A simple example will help illustrate the point. Consider the following base version of a routine, XYZ.

```
SUBROUTINE XYZ(A,B,C)
CALL SUB1(A,B)
C = A + B
WRITE(6,*)'hello',A,B,C
RETURN
END
```

Assume that two persons produce the following two versions of the subroutine XYZ. The first version is contained in the file XYZ1.SOR and the second version is contained in the file XYZ2.SOR.

```
SUBROUTINE XYZ(A,B,C,X,Y,Z)
CALL SUB1(A,B)
C = A + B
CALL SUB2(X,Y,Z)
WRITE(6,*)'hello',A,B,C
RETURN
END

SUBROUTINE XYZ(A,B,C,Z,Y,X)
CALL SUB1(A,B)
C = A + B
WRITE(6,*)'hello',A,B,C
Z = X + Y
WRITE(6,*)X,Y,Z
RETURN
END
```

To see the differences between these two routines use the command

VDIFF XYZ1.SOR XYZ2.SOR

to obtain

```
   XYZ1.FOR (08 May 1990 13:50:30)
            XYZ2.FOR (08 May 1990 13:34:30)
====================
+             1  |     SUBROUTINE XYZ(A,B,C,Z,Y,X)
-     1          |     SUBROUTINE XYZ(A,B,C,X,Y,Z)
      2       2  |     CALL SUB1(A,B)
      3       3  |     C = A + B
-     4          |     CALL SUB2(X,Y,Z)
      5       4  |     WRITE(6,*)'hello',A,B,C
+             5  |     Z = X + Y
+             6  |     WRITE(6,*)X,Y,Z
      6       7  |     RETURN
      7       8  |     END
====================
```

To merge the two versions together use the command

VMRG -P XYZ.FOR XYZ1.SOR XYZ2.SOR > XYZ.MRG

where '-P' and '> XYZ.MRG' indicate that the merged corrections are to be placed in the file XYZ.MRG, XYZ.SOR is the base file and XYZ1.SOR, XYZ2.SOR are the two versions to be merged. The merged file is given by

```
>>>>>>>>>>  XYZ1.FOR
      SUBROUTINE XYZ(A,B,C,X,Y,Z)
<<<<<<<<<<  XYZ2.FOR
      SUBROUTINE XYZ(A,B,C,Z,Y,X)
==========
      CALL SUB1(A,B)
      C = A + B
      CALL SUB2(X,Y,Z)
      WRITE(6,*)'hello',A,B,C
      Z = X + Y
      WRITE(6,*)X,Y,Z
      RETURN
      END
```

Note that the subroutine statement in the two modified files was modified by both developers. VMRG report this result by enclosing both version of the changes within the symbols >>>>>>>>>>>>>>>, <<<<<<<<<<<< and ==========. The other changes did not interfere with each other and were merged into one file. The resulting merged file can now be checked back in to the log file using the put command.

3.4 The control program DSOURC

As discussed in section 2.1, DSOURC is the module that controls most of the model calculations. SOLVE coordinates the time splitting technique, but most physical phenomena are accessed by DSOURC. It is split into several parts. First, the plume(s) are calculated. Then the fire model is done, depending on the type of fire. Then all other physical processes are included, natural and forced flow, convective and radiative heating and so forth. The module is tightly coupled with the solver and the physical routines. All routines from here on down are accessed many times per simulation run, so some thought should be given to the actual form of the routines that are to be included or changed. This portion of the model is the real numerical implementation, and the execution time is very sensitive to the actually coding of the software. What follows is an annotated form of the routine DSOURC. Extraneous comments have been left out in order to shorten it somewhat.

```
      SUBROUTINE DSOURC (TN, PDIF, C, D, PORC)

common blocks go here, also miscellaneous definitions used in the code, mostly temporary variables
```

fix local constants which are internal variables not set by the data copy routine

```
      XHALF = .50D0
      GAMMA1 = GAMMA - 1.0D0
      ONEBTA = GAMMA1 / GAMMA
      CV = CP / GAMMA
      XX0 = 0.0D0
      XX2 = 2.0D0
      DO 4 I = 1, N
         DO 1 K = UPPER, LOWER
            QF(K,I) = 0.0D0
            QFC(K,I) = 0.0D0
            QFR(K,I) = 0.0D0
    1    CONTINUE
         HEATVF(I) = 0.0D0
         HEATUP(I) = 0.0D0
         HEATLP(I) = 0.0D0
         EME(I) = 0.0D0
         EMP(I) = 0.0D0
         EMS(I) = 0.0D0
    4 CONTINUE
```

invoke the data copy procedure to set up the environment

```
      <<<<>>>>>>
```

calculate the species transport - only for species present - note that stport
is needed here to get the mass of each layer

```
      CALL STPORT (NETMAS, OLDMAS, NETFL, MASS, HWJ)
```

this is the pyrolysis rate for the fire, plus species production, assuming no
constraints

```
      CALL PYROLS (STIME, EMP(LFBO), ...)
```

divvy up the radiation from the lower plume to upper and lower portions

```
      CALL DISRAD(HR(LFBO)-Z(LFBO),HFOT,AW(UPPER),CHIRAD,VIEWUP,VIEWLW)
```

this is the actual convective heat release rate after subtracting off heating of the plume

```
      QHEATL = (QPYROL + CP*(TE-TL(LFBO))*EMP(LFBO))*(1-CHIRAD)
```

calculate the entrainment rate but constrain the actual amount of air that can
be entrained to that required to produce stable stratification this code
prevents the entrainment from exceeding that which would allow the plum the top
of the compartment

```
      CALL FIRPLM (QHEATL, MAX(0.,HR(LFBO)-HFOT-Z(LFBO)), EMP(LFBO),
     . EMS(LFBO), EME(LFBO), LFPOS)
      EME(LFBO) = MIN (EME(LFBO), QHEATL / ((TAMB(LFBO) + TG(LOWER))*RGAS))
      EMS(LFBO) = EMP(LFBO) + EME(LFBO)
```

constrain the species entrained into the plume based on actual available mass
in the lower layer

```
      IF (LFBT.GT.1) THEN
          DO 14 LSP = 1, 9
             DMASS(LSP) = MASS(LOWER,LFBO,LSP) * (EME(LFBO)/OLDMAS(LOWER,LFBO))
   14     CONTINUE
          EME(LFBO) = 0.0D0
          DO 15 LSP = 1, 9
```

```
            EME(LFBO) = EME(LFBO) + DMASS(LSP)
   15     CONTINUE
          EMS(LFBO) = EMP(LFBO) + EME(LFBO)
        ENDIF

now we start the actual calculation of source terms, first the fire is one of
two type, unconstrained(1) or constrained(2) - this first part only does the
lower plume, the upper plume and vent fires are done separately

        IF (LFBT.EQ.1) THEN

a type 1 fire uses no kinetics - we assume all fuel burns, regardless

          QPYROL = QHEATL / (1.0D0-CHIRAD)
          DO 8 LSP = 1, NS
             IF(.NOT.ACTIVS(LSP)) GO TO 8
             K = UPPER
             NETMAS(K,LFBO,LSP) = NETMAS(K,LFBO,LSP) + MFIRET(LSP)

add in the flow entrained by the plume(s)

             NEWNET = EME(LFBO) * MASS(LOWER,LFBO,LSP) /
     .                OLDMAS(LOWER,LFBO)
             NETMAS(UPPER,LFBO,LSP) = NETMAS(UPPER,LFBO,LSP) + NEWNET
             NETMAS(LOWER,LFBO,LSP) = NETMAS(LOWER,LFBO,LSP) - NEWNET
    8     CONTINUE

        ELSE

type 2 or later fire - include diffusion limited kinetics

          CALL CHEMIE (QPYROL, EMP(LFBO), EME(LFBO), NETFUEL, LFBO, LOWER,
     .                 NETMAS, OLDMAS, CCO2T, COCO2T, HCRATT,
     .                 OCRATT, CLFRAT, CNFRAT)

add in the species produced by the fire (see "allowed"); chemie has already
done the species production by burning within the plume, this is only a function
of pyrolysis

          DO 9 LSP = 1, NS
             IF (ACTIVS(LSP))
     .          NETMAS(UPPER,LFBO,LSP) = NETMAS(UPPER,LFBO,LSP)
     .                                   + MFIRET(LSP)
   9     CONTINUE
        ENDIF

we have finished the lower plume calculation, so sum up the heat release
here we have the energy balance for the lower layer burning or total burning
if this is an unconstrained fire

        QFR(UPPER,LFBO) = QPYROL * CHIRAD * VIEWUP
        QFR(LOWER,LFBO) = QPYROL * CHIRAD * VIEWLW
        QFC(UPPER,LFBO) = QPYROL * (1.0D0-CHIRAD)
        QF(UPPER,LFBO) =  QFC(UPPER,LFBO) + QFR(UPPER,LFBO)
        QF(LOWER,LFBO) =  QFC(LOWER,LFBO) + QFR(LOWER,LFBO)
        HEATLP(LFBO) = QF(UPPER,LFBO) + QF(LOWER,LFBO)

the following code is for a constrained fire only, that is one that can burn
in the upper layer, or in vents; if it is type one, then skip

        IF (LFBT.LE.1) GO TO 6

the heat release into the upper layer is from two sources, first the burning
of a plume in the upper layer, and second a vent fire, which deposits all
of its energy into the upper layer of an adjacent compartment
```

```
first do the plume in the upper layer; start with the fuel left over from
lower layer burning
umplm(ep),(es),and (ee) are the counterparts to emp, ems and  eme

      UPLMEP = MAX (XX0, EMP(LFBO)-NETFUEL)

check if anything to burn

      IF (UPLMEP.LE.XX0) GO TO 7

no view factor calculation in the upper layer - everything stays here

      QHEATU = HCOMBA * UPLMEP + QHEATL
      CALL FIRPLM (QHEATU, Z(LFBO), UPLMEP, UPLMES, UPLMEE, LFPOS)
      CALL CHEMIE (QPYROL, UPLMEP, UPLMEE, NETFUEL, LFBO, UPPER,
     .             NETMAS, OLDMAS, CCO2T, COCO2T, HCRATT,
     .             OCRATT, CLFRAT, CNFRAT)
      QFC(UPPER,LFBO) = QFC(UPPER,LFBO) + QPYROL
      QF(UPPER,LFBO) =  QFC(UPPER,LFBO) + QFR(UPPER,LFBO)
      HEATUP(LFBO) = QPYROL

now do the vent fires

note that sau to the ambient is very large (xlrg) so that there is not an
artificial constraint on the burning rate outside.  this is set in the model
during initialization

7     CONTINUE
      DO 5 I = 1, N
         DO 5 J = 1, N
            IF (NW(J,I).LE.0.OR.TU(J).LT.TGIGNT) GO TO 5
            SAUK = 0.0
            DO 2 K = 1, 4
               SAUK = SAUK + SAU(J,I,K)
    2       CONTINUE
            IF (SAUK.GT.0.0D0) THEN
               CALL CHEMIE (QPYROL, NETFL(J,I), SAUK, NETFUEL, I, LOWER,
     .                      NETMAS, OLDMAS, CCO2T, COCO2T, HCRATT,
     .                      OCRATT, CLFRAT, CNFRAT)
               HEATVF(I) = HEATVF(I) + QPYROL
               QFC(UPPER,I) = QFC(UPPER,I) + QPYROL
               QF(UPPER,I) = QFC(UPPER,I) + QFR(UPPER,I)
            ENDIF
    5 CONTINUE

now do all other source terms starting with the convective heat transfer

    6 DO 100 I = 1,NM1
         ACEILI    = AR(I)
         AUPPER    = (BR(I) + DR(I)) * Z(I) * 2.0D0
         AW(UPPER) = ACEILI + AUPPER
         AFLOOR    = AR(I)
         ALOWER    = MAX(XX0, (BR(I) + DR(I)) * (HR(I) - Z(I)) * XX2)
         AW(LOWER) = AFLOOR + ALOWER
         DO 11 J = 1, NWAL
            TW(J) = TWJ(J,I,1)
            EP(J) = EPW(J,I)
   11    CONTINUE
         TG(UPPER) = TU(I)
         TG(LOWER) = TL(I)
         QC(UPPER,I) = 0.0D0
         IF (SWITCH(1,I)) THEN
            CALL CONVEC(1, TG(UPPER), TW(1), ACEILI,QSCNV(1,I),
                        QC(UPPER,I))
```

```
          ELSE
             TW(1) = TG(UPPER)
             TWJ(1,I,1) = TG(UPPER)
          END IF
          IF (SWITCH(3,I)) THEN
             CALL CONVEC(3, TG(UPPER), TW(3), AUPPER,QSCNV(3,I), QC(UPPER,I))
          ELSE
             TW(3) = TG(UPPER)
             TWJ(3,I,1) = TG(UPPER)
          END IF
          QC(LOWER,I) = 0.0D0
          IF (SWITCH(2,I)) THEN
             CALL CONVEC(2, TG(LOWER), TW(2), AFLOOR,QSCNV(2,I), QC(LOWER,I))
          ELSE
             TWJ(2,I,1) = TG(LOWER)
             TW(2) = TG(LOWER)
          END IF
          IF (SWITCH(4,I)) THEN
             CALL CONVEC(4, TG(LOWER), TW(4), ALOWER,QSCNV(4,I), QC(LOWER,I))
          ELSE
             TW(4) = TG(LOWER)
             TWJ(4,I,1) = TG(LOWER)
          END IF

next is radiative heat transfer

          CALL FIRRAD (TW, TG, AW, AR(I), EP, XHALF, QSRADW, QR, I)

hydrogen chloride adsorption - note that this is a wall term and thus different
than normal species production and destruction

          IF (ACTIVS(6)) THEN
             IF (SWITCH(1,I)) THEN
                CALL HCLTRAN (UPPER,I,ACEILI,TU(I),TW(1),HWJDOT(1,I),1,NETMAS)
             END IF
             IF (SWITCH(3,I)) THEN
                CALL HCLTRAN (UPPER,I,AUPPER,TU(I),TW(3),HWJDOT(3,I),3,NETMAS)
             ENDIF
             IF (SWITCH(2,I)) THEN
                CALL HCLTRAN (LOWER,I,AFLOOR,TL(I),TW(2),HWJDOT(2,I),2,NETMAS)
             END IF
             IF (SWITCH(4,I)) THEN
                CALL HCLTRAN (LOWER,I,ALOWER,TL(I),TW(4),HWJDOT(4,I),4,NETMAS)
             END IF
          ENDIF

once we do smoke agglomeration, it will be similar to the hcl above

       IF (ACTIVS(9)) THEN
       ENDIF

this is the end of the "other source terms" calculation

  100 CONTINUE

now we do flow - first straight flow between compartments, upper to upper,
lower to lower, both ways

       DO 203 I = 1, N
       DO 203 J = 1, N
          IF (NW(I,J).EQ.0) GO TO 203
          DO 202 K = 1, 4
             IF (IAND(FRMASK(K),NW(I,J)).NE.0)  CALL FLOW (I, J, K, ...)
  202     CONTINUE
  203 CONTINUE
```

```
now entrained flow - the result sau and asl do not include the pseudo plumes
themselves - first entrainment into the upper layer

      DO 120 I = 1, N
      DO 120 J = 1, N
         IF (NW(J,I).EQ.0) GO TO 120
         IF (J.EQ.N) THEN
            TU(J) = ETA(I)
            TL(J) = ETA(I)
         ELSE IF (I.EQ.N) THEN
            TU(I) = ETA(J)
            TL(I) = ETA(J)
         ENDIF
loop over vents between compartments (k =1->4)
         DO 121 K = 1, 4
            IF (IAND(FRMASK(K),NW(I,J)).EQ.0) GO TO 121
            IF (SA(J,I,K).LT.0.0001D0.OR.TU(J).LE.TL(I)) THEN
               SAU(J,I,K) = 0.0D0
            ELSE
zd is initialized to xlrg in the model for the outside - see above explanation
               ZD = MAX(XX0,(HRP(I)-Z(I))-(HRP(J)-Z(J)))
               CALL ENTRFL(TU(J), TL(I), SA(J,I,K), ZD, SAU(J,I,K))
            ENDIF
  121    CONTINUE
  120 CONTINUE

now into the lower layer

      DO 122 I = 1, N
      DO 122 J = 1, N
         IF (NW(J,I).EQ.0) GO TO 122
         IF (J.EQ.N) THEN
            TU(J) = ETA(I)
            TL(J) = ETA(I)
         ELSE IF (I.EQ.N) THEN
            TU(I) = ETA(J)
            TL(I) = ETA(J)
         ENDIF
         DO 123 K = 1, 4
            IF (IAND(FRMASK(K),NW(I,J)).EQ.0) GO TO 123
            IF (AS(J,I,K).LT.0.0001D0.OR.TU(I).LE.TL(J)) THEN
               ASL(J,I,K) = 0.0D0
            ELSE
               IF (J.LT.N.AND.I.LT.N) THEN
                  ZD = MAX(XX0,(HRP(J)-Z(J))-(HRP(I)-Z(I)))
               ELSE
                  IJ = MIN(I,J)
                  JI = MAX(J,I)
                  ZD = MAX(XX0,HH(IJ,JI,K)-(HR(IJ)-Z(IJ)))
               END IF
               CALL ENTRFL(TU(I), TL(J), AS(J,I,K), ZD, ASL(J,I,K))
               ASL(J,I,K) = MAX(XX0, ASL(J,I,K)-MINMAS)
            ENDIF
  123    CONTINUE
  122 CONTINUE

sort into source terms for the solver, i for compartment, j for source, k for vent

      DO 10 I=1,NM1
         OPRES = 1.0D0 / (P(I)+POFSET)
         VU = P(I+N2)
         VL = MAX(VR(I)-VU,PMIN(I+N2))

set      SPIN,SMNET,SLIN,STIN,SUNET,SLNET  to 0.0D0
```

```
         GV = CNVG(1) * GAMMA1 / VR(I)
         GTU = ONEBTA * OPRES * TU(I) / VU
         GTL = ONEBTA * OPRES * TL(I) / VL

natural flow

         DO 20 J=1,N
            IF (NW(J,I).EQ.0) GO TO 20

            IF (J.EQ.N) THEN
               TU(J) = ETA(I)
               TL(J) = ETA(I)
            ENDIF

mass flux definitions for all cases

            DO 30 K = 1, 4

if no flow, then forget it

               IF (IAND(FRMASK(K),NW(I,J)).EQ.0) GO TO 30

               SSIN = SS(J,I,K)
               SSOT = SS(I,J,K)
               SAIN = SA(J,I,K)
               SAOT = SA(I,J,K)
               ASIN = AS(J,I,K)
               ASOT = AS(I,J,K)
               AAIN = AA(J,I,K)
               AAOT = AA(I,J,K)

the following rules are discussed in the references, but take care of the
problems due to the natural stack effect

case 1 - tu(i) > tl(i) > tu(j) > tl(j)

               IF (TL(I).GT.TU(J)) THEN
                  INULU = 0.0D0
                  INULL = 0.0D0
                  INLLU = SSIN + SAIN
                  INLLL = ASIN + AAIN

case 2 - tu(j) > tu(i) > tl(i) - this is the opposite of the first case

               ELSE IF (TL(J).GT.TU(I)) THEN
                  INULU = SSIN + SAIN
                  INULL = ASIN + AAIN
                  INLLU = 0.0D0
                  INLLL = 0.0D0

case 3 - otherwise

               ELSE
                  INULU = SSIN + SAIN
                  INULL = 0.0D0
                  INLLU = 0.0D0
                  INLLL = ASIN + AAIN
               ENDIF

now add up the mass flows into the proper parts

               INUL = INULU + INULL
               INLL = INLLU + INLLL
               OTUL = SSOT + SAOT
               OTLL = ASOT + AAOT
```

```
               SMNET = SMNET + (INUL-OTUL) + (INLL-OTLL)
               SUNET = SUNET + (INUL-OTUL) + (SAU(J,I,K)-ASL(J,I,K))
               SLNET = SLNET + (INLL-OTLL) + (ASL(J,I,K)-SAU(J,I,K))

now we can do enthalpy flow

pressure
               SPIN = SPIN
     .                + (TU(I)-TAMB(I)) * (INUL-OTUL)
     .                + (TL(I)-TAMB(I)) * (INLL-OTLL)
     .                + (TU(J)-TU(I))   * INULU
     .                + (TL(J)-TU(I))   * INULL
     .                + (TL(J)-TU(I))   * INLLU
     .                + (TL(J)-TL(I))   * INLLL

upper layer temperature
               STIN = STIN
     .                + (TU(J)-TU(I))   * INULU
     .                + (TL(J)-TU(I))   * INULL
     .                + (TL(I)-TU(I))   * SAU(J,I,K)

lower layer temperature
               SLIN = SLIN
     .                + (TL(J)-TU(I))   * INLLU
     .                + (TL(J)-TL(I))   * INLLL
     .                + (TU(I)-TL(I))   * ASL(J,I,K)
   30       CONTINUE

this concludes the "natural" mass flow

   20    CONTINUE

add in the contribution from mechanical ventilation - forced flow

         DO 50 J = 1, NEXT
            IF (HVNODE(1,J).EQ.I) THEN
               SMNET = SMNET + HVEFLO(UPPER,J) + HVEFLO(LOWER,J)
               SUNET = SUNET + HVEFLO(UPPER,J)
               SLNET = SLNET +                   HVEFLO(LOWER,J)
               SPIN = SPIN + (HVEXTT(J)-TU(I))*HVEFLO(UPPER,J)
     .                     + (HVEXTT(J)-TL(I))*HVEFLO(LOWER,J)
               STIN = STIN + (HVEXTT(J)-TU(I))*MAX(XXO,HVEFLO(UPPER,J))
               SLIN = SLIN + (HVEXTT(J)-TL(I))*MAX(XXO,HVEFLO(LOWER,J))
            ENDIF
   50    CONTINUE

finally, include the plume contribution(s)

         SUNET = SUNET + EMS(I)
         SMNET = SMNET + EMP(I)
         SLNET = SLNET - EME(I)

now put it all together for the solver

         QUI = QF(UPPER,I) + QR(UPPER,I) + QC(UPPER,I)
         QLI = QF(LOWER,I) + QR(LOWER,I) + QC(LOWER,I)

         C(I) = (QUI + QLI + CP*(SPIN+EMP(I)*(TE-TAMB(I)))
     .          + CV*TAMB(I)*SMNET) * GV
         D(I) = 0.0D0
         C(I+N) = (QUI + CP*(STIN+EMP(I)*(TE-TU(I)) +
     .             EME(I)*(TL(I)-TU(I)))
     .        - RGAS*(TU(I)-TAMB(I))*SUNET + VU*C(I)) * GTU
         D(I+N) = 0.0D0
```

```
            C(I+N2) = OPRES * (RGAS*TU(I)*SUNET - VU*C(I))
     .                + C(I+N)/TU(I)*VU
            D(I+N2) = 0.0D0
            C(I+N3) =(QLI + CP*SLIN  - RGAS*(TAMB(I)-TL(I))*SLNET +
     .                  VL*C(I)) * GTL
            D(I+N3) = 0.0D0
     10 CONTINUE

   done!!! repack the species for the solver - only the basic variables are in
   the proper format - first "normal species" n2,o2,... then hcl deposition, which
   acts like a species, that is, the hwjdot from hcltran is a loss term, then
   smoke density which is an agglomeration, deposition effect

         ISOF = N4
         DO 40 LSP = 1, NS
            IF(.NOT.ACTIVS(LSP)) GO TO 40
            DO 41 I = 1, NM1
            DO 41 K = UPPER, LOWER
               ISOF = ISOF + 1
               C(ISOF) = NETMAS(K,I,LSP)
               D(ISOF) = 0.0D0
     41    CONTINUE
     40 CONTINUE

         IF (ACTIVS(6)) THEN
            DO 42 I = 1, NM1
            DO 42 K = 1, NWAL
               ISOF = ISOF + 1
               C(ISOF) = HWJDOT(K,I)
               D(ISOF) = 0.0D0
     42    CONTINUE
         ENDIF

         IF (ACTIVS(9)) THEN
         ENDIF

   go home

         RETURN
         END
```

4. PROTOCOL AND INITIAL SERVICES

Information is passed between the various subroutines and main modules by the use of files and common blocks. The model has several common blocks associated with it. Of interest to most programmers is the way the data are used in the model. However, there is some other information which is made available and will be discussed first.

Each main module calls a set of routines which set up the physical environment for the model. The routines are READOP and OPENSHEL. They perform a number of housekeeping tasks. This information is related to the environment (computer platform) on which the model is running.

First, the command line is interpreted, providing file names and options. Up to two file names are available. The first entry is either an input file, or a configuration file. If it is an input file, then the name is provided in the variable "NNFILE" in the shell common block. It will not be opened. When a file is opened for input, the unit number IOFILI should be assigned. If there is a valid output file, its name is provided in the variable "OUTFILE." It will be opened, and the unit number is IOFILO. There is a default configuration file CFAST.CF which will be read unless an alternative is provided on the command line. A configuration file name can be provided in place of the input file. In this case, the input file will be fetched from the variable DFILE from within the configuration file. In any case, the current configuration file will be named in CONFIG. Most modules will not run without the configuration file.

Second, the options are stored in the shell common blocks. There are currently five options available. All five are available to all modules. They are specified on the command line by

- (or /) option.

An example is the option to prevent the header from printing. This would be

-N.

The five which are presently read and decoded are

1. Report type (Rnn)
2. Graphics card (Gt:nn)
3. No header (N)
4. Turn on error logging (L)
5. Pass an environment file (Ffilename) - *only for the shell to pass an environment file*

These options are read and interpreted by the routine READOP.

The data file is opened by the input routine, NPUTP. It is closed by this routine after all data has been read. There should be no units assigned, opened or closed while data retrieval is in process. Thus data entry should be done within the scope of NPUTP or NPUTQ only by these routines. Any data can be retrieved from the data files by NPUTQ. If subsidiary information is needed, then a reference file name should be read and stored in a variable kept in the PARAMS common block (or unlabeled common), and data fetches made after the *second* exit from NPUTP. This precludes initialization of such data during the geometry and fire specification process. The thermal properties are retrieved by the initialization routines at this point (after the restart return).

New key words that are to be added to the data file are placed in NPUTQ. It is important to follow the protocol as laid out therein, so that consistency checks can be performed on the data. Any physical initialization that needs to be done should be included in NPUTP, after the call to LOADIN. At this point, the model is completely set up. The only data that is not done are the total masses of the upper and lower layers. This is deferred to the originating routine. The name of the primary data file is in the variable NNFILE, but is open at this point, and pointing to the end of NNFILE. NPUTP can be referenced twice if a restart has been requested. In each case, NNFILE is closed prior to exiting.

In the process of inserting key words into NPUTQ, one will note that there are two case statements (computed GO TO) for the key words. The first is for a normal start, and the second is for a restart. In the latter case, some variables are not, and should not, be valid. An example of the difference: the fire specification can change, but it makes no sense to change the physical layout.

The next consideration is the setup provided for the physical system. Initialization is done by CFAST, or INITFS in the case of the data editor. Both preset memory, and call the routine NPUTP. NPUTP does the actual physical initialization. It in turn calls the routine NPUTQ which reads the data files. Subsequently, the geometry, species and graphics descriptors are set. Finally the environment is set by CFAST or INITFS. This includes reading the thermophysical properties and assigning them to the correct boundary, and all other auxiliary files as necessary. It is at this point that all the data files are closed, and the unit IOFILI is available. It should be closed, opened to the appropriate file, and *subsequently closed.*

There are several logical switches that are set, based on the problem to be solved. The two most common are ACTIVS and SWITCH. The former is for active species. The latter serves two purposes, for active conduction and for miscellaneous parameters.

If a species is being computed, for whatever reason, then ACTIVS will be TRUE, otherwise it will be FALSE. This parameter is dimensioned to NS, the number of species which CFAST will follow. They are documented in STPORT. The order is

INDEX	SPECIES	APPLICABLE KEY WORD
1	Nitrogen	none
2	Oxygen	O2
3	Carbon dioxide	none
4	Carbon monoxide	CO
5	Water vapor	none but HCR is related
6	Hydrogen chloride	HCL
7	Unburned hydrocarbons	none
8	Hydrogen cyanide	HCN
9	Soot	OD
10	Concentration time dose	CT (not a species)

For each species that is tracked, the variable ACTIVS(i) is set to true. There are two types of action that hinge on the setting of this variable. The first is in the availability and display of species information. The second is in the packing used in preparing the source terms for, and extracting them from the solver. The details of this activity are in the section on the **data copy** and DSOURC routines.

The variable SWITCH is used in two places. The first is to specify which boundaries in which compartments can conduct heat. The parameter is set in NPUTQ, but verified by NPUTT. It can be set in NPUTQ if specified in the data file, but subsequently turned off by NPUTT if the name of the boundary can not be found in the thermophysical database. When the primary model is running, it will terminate if this latter condition is found, whereas the data editor will distinguish between the boundary being considered adiabatic with the name "OFF" and not found by "NONE." SWITCH is dimensioned NWAL by NR.

Since conduction is only allowed for NR-1 compartments, the last column can be used for miscellaneous variables. Once again, the default is false, but if the appropriate key word has been set, then the variable will be set to true.

(1,NR) - print the flow field and species - set in NPUTP; used only by the main model
(2,NR) - use the semi infinite slab approximation for heat loss by conduction
(3,NR) - CNVG(1) has been set - used in DSOURC for the pressure damping
(4,NR) - CNVG(2) has been set - changes error tolerance in the solver DIFEQ

The order of initialization is important. This is particularly true because of the caveat above that input/output units should not be assigned during the primary initialization. First, the main routine, CFAST or INITFS, initialize memory, and some physical constants such as the gravitational constant. The main data file is then opened. If one can not be found, then the model quits. The next step is to call NPUTP with the restart parameter of ISRSTR=1.

NPUTP reads the header line to check for a correct file, then calls NPUTQ, with the ISRSTR=1. NPUTQ does all of the actual data entry, *via* unit IOFILI. After control is returned to NPUTP, physical initialization is done, for example setting the atmospheric ambient, calculating the volume of the compartments, and so forth. Then the graphics descriptors are read by LOADIN. These processes occur whether or not a restart will be done. Then control is returned to the main module. Some additional processing takes place to set the species of the ambient environment. If a restart has been requested, the appropriate history file is read for the requested interval. NPUTP rewinds the input file and once again calls NPUTQ, with ISRSTR=2. At this point there are some differences. Within NPUTQ, the case statement (discussed above) prevents some parameters from being reset.

Units are specific to the operating system. There are two general input/output units named IOFILI and IOFILO. In the current implementation they are numbered 1 and 6 respectively. There are additional units as follows:

1) ISRSTR tells us if this is a normal input file, or a restart
   - 0 => MOST DATA HAS BEEN SET - DO INITIALIZATION ONLY
   - 1 => IMPLIES A NORMAL READ, WITH OPEN
   - 2 => IMPLIES AN UPDATE AFTER RESTART

2) files used
   - 1 => CONFIGURATION FILE, PRIMARY DATA FILE AND DATA BASES - OPENED AND CLOSE BY EACH ROUTINE
   - 2 => HELP FILES
   - 3 => LOG FILE - OPEN ALL THE TIME
   - 9 => RESTART AND DUMP FILE I/O & FONT FILES - BOTH ARE OPEN INTERMITTENTLY

Initialization is complete, and we let the simulation begin.

5. DESCRIPTION OF THE DATA FILES USED BY CFAST

There are two types of files used within the model. Most files are in ASCII format, and can be listed with a "cat" or "type" command. The one file that is binary is the history file used for saving the time histories of the variables. This latter file is not portable, whereas the ASCII data file can be moved across platforms. The only comment on the binary file, is that the order of variables is that in the common block "moco1a." It is written by the routine WRITEOT and read for restarts and plotting by DREADIN.

The remaining files are ASCII. They are

1. the main data file,
2. the thermophysical data file,
3. a file containing multiple objects,
4. a configuration file.

In addition, the model writes results to IOFILO, and this can be a file and printed or examined later. It too will be in ASCII format. The other files shown in the configuration file are not utilized at present. They are for future enhancement. In particular, the geometry file will be for connecting walls for heat transfer between compartments, and the partition file will be serve a similar purpose for destructible walls.

1. The main data file is free format. A key word must begin a line, and only the first five characters are significant. The model will read and write 256 byte records, which is generally long enough to keep all parameters. The detailed form of the file is given in Appendix A.

2. The format of the thermophysical data (default is THERMAL.DAT) file is

```
Name,  conductivity, specific heat, density, thickness, emissivity, and seven
coefficients for hydrogen chloride deposition, reference [2].
```

There can be up to NTHMX of these entries. It too is free format.

3. For multiple objects, there is a separate file (default is OBJECTS.DAT) which contains data for multiple objects. In the simplest case, the data appears very similar to the specified fire within the model. There are entries for heat release, *etc.* However, there are also entries for total mass, size, and so forth. The present version of the model does not use these entries, but they are provided since that information will be necessary for including self consistent burning objects. The format is

```
 1. Name up to 8 characters
 2. Type (1->4), position (1->3), (surface temperature & flux for ignition), total mass, gmw,
volitization temperature

<<< the following are identical to the specified fire >>>

 3. Time interval (1->lfmax)
 4. Pyrolysis rate (1->lfmax+1)
 5. Heat release rate (1->lfmax+1)
 6. Area of the fire (1->lfmax+1)
 7. Height of the base of the flame (1->lfmax+1)
 8. Ratio of co to co sub 2 in burning (1->lfmax+1)
 9. Ratio of soot to co sub 2 in burning (1->lfmax+1)
10. Ratio of hydrogen to carbon in the fuel (1->lfmax+1)
11. Ratio of oxygen to carbon in the fuel (1->lfmax+1)
12. Factor for concentration time dose calculation (1->lfmax+1)
13. Ratio of hydrogen cyanide to fuel pyrolysis rate (1->lfmax+1)
14. Ratio of hydrogen chloride to fuel pyrolysis rate (1->lfmax+1)
```

4. The configuration file has the following format

```
 1. heading key ($$CF$$)
 2. version(i4), colors (11xi3), units (7xi3), remote file(i3), advanced features(l1)
 4. thermal datafile (a60)
 5. last written file from data editor (a60)
 6. geometry file (compartments,...) (a60)
 7. other objects file (a60)
 8. partition properties file (a60)
 9. path for executable, databases, ... (a64)
10. path for data - if blank use current path (a64)
```

The format used by the "read" statements is shown in parenthesis.

Although this file is in the ASCII format, it should not be edited directly. The routine READCF which reads in the configuration file performs only rudimentary checks on the information. Since path names, file names and other crucial data are specified, the wrong information can cause disastrous results. If changes need to be made, then the pair of functions, READCF, and MAKECF should be altered. The former reads the file, and the latter writes the file. There are corresponding read/write lines in the two routines. For changes made in one, there MUST be corresponding changes made in the other. This will then keep the file format consistent.

APPENDIX A : PARAMETERS READ BY "NPUTQ"

The computer model requires a description of the problem to be solved. The following description is for the input data used by the model. In general, the order of the data is not important. The one exception to this is the first line which specifies the version number and gives the data file a title.

The data are grouped as

- Version and title (A.1)
- Time specification (A.2)
- Ambient conditions (A.3)
- Floor plan data (A.4)
- Connections (A.5)
- Thermophysical properties of the enclosing surfaces (A.6)
- Fire specifications (A.7)
- Species production (A.8)
- Files (A.9)
- Graphics specification (A.10)
- Mechanical ventilation (A.11)
- Miscellaneous (A.12).

The number of lines in a given data set will vary depending for example on the number of openings or the number of species tracked. A sample input data file is given in Appendix A. A number of parameters such as heat transfer and flow coefficients have been set within the program as constants. Please refer to the section on source terms to ascertain the values for these parameters.

Each line of the input data file begins with a key word which identifies the type of data on the line. The key words which are currently available are

CEILI	specify name of ceiling descriptor(s)	(N)
CHEMI	miscellaneous parameters for kinetics	(5)
CO	CO/CO_2 mass ratio	(lfmax+1)
CT	fraction of fuel which is toxic	(lfmax+1)
CVENT	opening/closing parameter	(lfmax + 4)
DEPTH	depth of compartments	(N)
DUMPR	specify a file name for saving time histories	(1)
EAMB	external ambient	(3)
FAREA	area of the base of the fire	(lfmax+1)
FHIGH	height of the base of the fire	(lfmax+1)
FLOOR	specify the name of floor property descriptor(s)	(N)
FMASS	pyrolysis rate	(lfmax+1)

FQDOT	heat release rate	(lfmax+1)
FTIME	length of time intervals	(lfmax)
HCL	hcl/pyrolysis mass ratio	(lfmax+1)
HCN	hcn/pyrolysis mass ratio	(lfmax+1)
HCR	hydrogen/carbon mass ration of the fuel	(lfmax+1)
HEIGH	interior height of a compartment	(N)
HI/F	absolute height of the floor of a compartment	(N)
HVENT	specify vent which connect compartments horizontally	(7)
INELV	specify **interior** node elevations (for ventilation ducts)	(2 x # of interior nodes)
INTER	initial height of the upper/lower interface	(2)
LFBO	compartment of fire origin	(1)
LFBT	type of fire	(1)
LFMAX	number of time intervals	(1)
LFPOS	position of the fire in the compartment	(1)
MVDCT	describe a piece of (circular) duct work	(9)
MVFAN	give the pressure - flow relationship for a fan	(3 to 8)
MVOPN	Specify an opening between a compartment and ventilation system	(5)
OD	C/CO_2 mass ratio	(lfmax+1)
O2	ratio of oxygen to carbon in the fuel	(lfmax+1)
RESTR	specify a restart file	(2)
TAMB	ambient inside the structure	(3)
TIMES	time step control of the output	(5)
VVENT	specify a vent which connects compartments vertically	(3)
VERSN	version number and title	(fixed format 2)
WALLS	specify the name of wall property descriptor(s)	(N)
WIDTH	width of the compartments	(N)
WIND	scaling rule for wind effects	(3)

The number in parenthesis is the maximum number of entries for that line. "N" represents the number of compartments being modeled and "lfmax" is the number of time intervals used to describe the fire, detailed below in section A.7. The outside (ambient) is designated by one more than the number of compartments, N+1. So a three compartment model would refer to the outside as compartment four.

Each line of input consists of a label followed by one or more alphanumeric parameters associated with that input label. The label must always begin in the first space of the line and be in capital letters. Following the label, the values may start in any column and all values must be separated by either a comma or a space. Values may contain decimal points if needed or desired. They are not required. Units are standard SI units. Most parameters have default values which can be utilized by omitting the appropriate line. These will be indicated in the discussion. The maximum line length is 128 characters, so all data for each key word must fit in this number of characters. For each entry which requires more than one type of data, the first entry under the column "parameter" indicates the number of data required.

A.1 Version and Title

This line must be the first line in the file. It is the line that CFAST keys on to determine whether it has a correct data file. The format is fixed, that is the data must appear in the columns specified in the text.

Label	Parameter	Comments
VERSN	(2)	The VERSN line is a required input.
	Version Number	The version number parameter specifies the version of CFAST for which the input data file was prepared. Normally, this would be 18. It must be in columns 8-9.
	Title	The title is optional and may consist of letters, numbers, and/or symbols that start in column 11 and may be up to 50 characters. It permits the user to uniquely label each run.

Example:

```
VERSN   18 Simulation for Building XYZ
```

A.2 Time Specification

Label	Parameter	Comments	Units
TIMES	(5)	The TIMES line is required data.	
	Simulation Time	Simulation time is the length of time over which the simulation takes place. The maximum value for this input is 86400 seconds (1 day). The simulation time parameter is required.	s

Print Interval	The print interval is the time interval between each printing of the output values. If omitted or less than or equal to zero, no printing of the output values will occur.	s
Dump Interval	The dump interval is the time interval between each writing of the output to the history file. The history file stores all of the output of the model at the specified interval in a format which can be efficiently retrieved for use by other programs. Section A.9 provides details of the history file. A zero must be used if no dump file is to be used. There is a maximum of 50 intervals allowed. If the choice of this parameter would yield more than 50 writes, it is adjusted so that this limit is not exceeded.	s
Display Interval	The display interval is the time interval between each graphical display of the output as specified in the graphics specification, section A.10. If omitted, no graphical display will occur. There is a limit on the display of 50 for the microcomputer versions and 100 for the mainframe versions of FAST. This parameter is not adjusted; rather graphs will be truncated to the first 50 or 100 points, respectively.	s
Copy Count	Copy count is the number of copies of each graphical display to be made on the selected hard copy device as specified in the graphics specification, section 2.9. If omitted, a value of zero (no copies) is assumed.	

Examples:

```
TIMES  360  0  0
TIMES  360 10 30
TIMES  900    30    10    10    0
```

In the first example, a simulation time of 360 s is specified. The output values will not be printed or stored in a history file. No graphical display of the output will occur. In the second example, a 360 s simulation with printed output every 10 s and output to a history file every 30 s is specified. No graphical display of the output values will be generated. In the third example, all parameters are specified. A 900 s simulation with printed output every 30 s, output to a history file every 10 s and a graphical display with no copies will occur every 10 s.

Note the free field format of these parameters -multiple spaces between parameters are permitted.

A.3 Ambient Conditions

The ambient conditions section of the input data allows the user to specify the temperature and pressure and station elevation of the ambient atmosphere, as well as the absolute wind pressure to which the structure is subjected. There is an ambient for the interior and for the exterior of the building. The key word for the interior of the building is TAMB and for the exterior of the building is EAMB. The form is the same for both. The key word for the wind information is WIND. The wind modification is applied only to the vents which lead to the exterior. Pressure interior to a structure is calculated simply as a lapse rate based on the NOAA tables [17]. For the exterior, the nominal pressure is modified by

$$\delta(p) = C_w \rho V^2 \quad \text{where} \quad V = V_w \left(\frac{H_i}{H_w}\right)^{p_w}.$$

This modification is applied to the vents which lead to the exterior ambient. The pressure change calculated above is modified by the wind coefficient for each vent. This coefficient, which can vary from -1.0 to +1.0, nominally from -0.8 to +0.8, determines whether the vent is facing away from or into the wind. The pressure change is multiplied by the vent wind coefficient and added to the external ambient for each vent which is connected to the outside.

Label	Parameter	Comments	
TAMB or EAMB	(3)	These data are optional.	
	Ambient Temperature	Ambient temperature is the temperature of the ambient atmosphere. Default is 300.	K
	Ambient Pressure	The ambient pressure is the pressure of the ambient atmosphere. Default is 101300.	Pa

	Station Elevation	The station elevation is the elevation of the point at which the ambient pressure and temperature (see above) are measured. The reference point for the elevation, pressure and temperature must be consistent. This is the reference datum for calculating the density of the atmosphere as well as the temperature and pressure inside and outside of the building as a function of height. Default is 0.	m
WIND	(3)	This line is optional.	
	Wind Speed	Wind speed at the reference elevation. The default is 0.	m/s
	Reference Height	Height at which the reference wind speed is measured. The default is 10 m.	m
	Lapse Rate Coefficient	The power law used to calculate the wind speed as a function of height. The default is 0.16.	

The choice for the station elevation, temperature and pressure must be consistent. Outside of that limitation, the choice is arbitrary. It is often convenient to choose the base of a structure to be at zero height and then reference the height of the building with respect to that height. The temperature and pressure must then be measured at that position. Another possible choice would be the pressure and temperature at sea level, with the building elevations then given with respect to mean sea level. This is also acceptable, but somewhat more tedious in specifying the construction of a building. Either of the these choices works though because consistent data for temperature and pressure are available from the Weather Service for either case.

Examples:

```
TAMB 300
TAMB 288 101000 200.
```

The first example sets the ambient temperature to 300 K, but leaves the ambient pressure at 101300 and the reference elevation at 0 m. The second specifies a temperature of 15 C at 200 m and a pressure of 101000 Pa. In both of these cases the external ambient is set to the same values. An example of different inside and outside values is a warm building in a winter setting and might be described as

```
TAMB  288 101305 0.0
EAMB  270 101315 0.0
```

A.4 Floor Plan Data

The floor plan data section allows the user to portray the geometry of the structure being modeled. The size and location of every room in the structure MUST be described. The maximum number of rooms is dependent upon the local implementation of FAST. Usually a total of 10 rooms (plus the outdoors) is available for a single simulation. For the PC versions, a maximum of six compartments (plus the outdoors) is allowed. The structure of the data is such that the compartments are described as entities, and then connected in appropriate ways. It is thus possible to have a set of rooms which can be configured in a variety of ways. In order to specify the geometry of a building, it is necessary to give its physical characteristics. Thus the lines labelled HI/F, WIDTH, DEPTH AND HEIGH are all required. Each of these lines requires "N" data entries, that is one for each compartment.

Label	Parameter	Comments	Units
HI/F	**Floor Height**	The floor height is the height of the floor of each room with respect to station elevation specified by the TAMB parameter. The reference point must be the same for all elevations in the input data. The number of values on the line must equal the number of rooms in the simulation.	m
WIDTH	**Room Width**	Room width specifies the width of the room. The number of values on the line must equal the number of rooms in the simulation.	m
DEPTH	**Room Depth**	Room depth specifies the depth of the room. The number of values on the line must equal the number of rooms in the simulation.	m
HEIGH	**Room Height**	Room Height specifies the height of the room. The number of values on the line must equal the number of rooms in the simulation.	m

Example:

```
HI/F   0.0   0.0   0.0
WIDTH  6.1   4.6   4.6
DEPTH  9.1  14.3   4.3
HEIGH  3.6   2.4   2.4
```

This floor plan data specifies the sizes for a three room simulation with rooms sizes of 6.1 x 9.1 x 3.6 m, 4.6 x 14.3 x 2.4 m, and 4.6 x 4.3 x 2.4 m, respectively. All rooms are at the same elevation at a reference height of 0.0 m.

A.5 Connections

The connections section of the input data file describes any horizontal or vertical vents between rooms in the structure. These may include doors between rooms in the structure, windows in the rooms (between rooms or to the outdoors), or vertical openings between floors of the structure. Openings to the outside are included as openings to the room with a number one greater than the number of rooms described in the floor plan data section. Doors, windows, and the like are called horizontal vents because the direction of the vent, or vent connection, is in the horizontal direction. The key word is HVENT. Horizontal vents may be opened or closed during the fire with the use of the CVENT key word. For vertical vents, such as scuddles, the key word is VVENT; at present there is not an equivalent mechanism for opening or closing the vertical vents. The form for horizontal and vertical vents is necessarily different.

Label	Parameter	Comments	Units
HVENT	(7)	Required to specify connections between compartments. No openings prevents flow. Each HVENT line in the input file describes one horizontal vent between rooms in the structure (or between a room and the outdoors). The first six entries on each line are required. There is an **optional** seventh parameter to specify a wind coefficient.	
	First Room	The first room is simply the first connection.	
	Second Room	The second room is the room number to which the first room is connected. The order has one significance. The height of the sill and soffit are with respect to the first compartment specified.	
	Vent Number	There can be as many as four vents between any two compartments. This number specifies which vent is being described. It can range from one to four.	

	Width	The width of the opening.	m
	Soffit	Position of the top of the opening above the floor of the room number specified as the first room.	m
	Sill	Sill height is the height of the bottom of the opening above the floor of the room number specified as the first room.	m
	Wind	The wind coefficient is the cosine of the angle between the wind vector and the vent opening. This applies only to vents which connect to the outside ambient (specified with EAMB). The range of values is -1.0 to +1.0. If omitted, the value defaults to zero.	
VVENT	(3)	Required to specify a vertical connection between compartments. Each VVENT line in the input file describes one vertical vent between rooms in the structure (or between a room and the outdoors). There are three parameters, the connected compartments, and the **effective** area of the vent.	
	First Room	The first room is simply the first connection.	
	Second Room	The second room is the room number to which the first room is connected. The order has one significance. The height of the sill and soffit are with respect to the first compartment specified.	
	Area	This is the effective area of the opening. For a hole, it would be the actual opening. For a diffuser, then the effective area will be somewhat less than the geometrical size of the opening.	

Examples:

```
HVENT 1  2  1 1.1  2.1  0.0
HVENT 1  3  1 1.1  2.1  0.0
HVENT 2  4  1 1.3  2.1  0.6
VVENT 1  3  3.0
```

Assuming the three room structure as described in the floor plan data section, the above examples describe two openings 1.1 x 1.5 m between rooms 1 and 2 and between rooms 1 and 3. An 1.3 x 2.1 m opening between room 2 and the outside (room 4 for a three room simulation) is raised 0.6 m off the floor of room 2.

```
HVENT 2  4  2 1.3  2.1  0.6 1.0
```

This specifies vent #2 between compartment (2) and the outside, with a wind coefficient of 1.0, which implies that the vent is facing directly into the wind.

CVENT is a parameter which is used to open and close vents. It multiples the width in the vent flow calculation. The default is 1.0 which is a fully open vent. A value of 0.5 would specify a vent which was halfway open.

Label	Parameter	Comments	Units
CVENT	(LFMAX+4)	Specify closing value. Each CVENT line in the input file describes one horizontal vent between rooms in the structure (or between a room and the outdoors).	
	First Room	The first compartment.	
	Second Room	The second room is the room number to which the first room is connected.	
	Vent Number	This number specifies which vent is being described. It can range from one to four. These parameters correspond to the first three parameters in HVENT.	
	Width (LFMAX+1)	Fraction that the vent is open. This applies to the width only. The sill and soffit are not changed.	%

CVENT has a form similar to HVENT but in addition contains the opening data. The additional data is in the same form as all the time dependent specifications, namely a value for each endpoint in the heat release curve. The form is

CVENT C#1 C#2 V# x x x x,...

By way of example, the default value for CVENT for the example show above with LFMAX=5 would be

```
CVENT 1 2 1 1.0 1.0 1.0 1.0 1.0 1.0
```

and would specify that the first vent between compartments (1) and (2) would be open at all times. Another example would be

```
CVENT 1 3 1 0.5 0.5 0.5 0.5 0.5 0.5
```

and would specify that the first vent between compartments (1) and (3) would be half open all of the time. These fractions refer to the width given in the HVENT specification and for the cases above would be 1.1 m.

A.6 Thermophysical Properties of Enclosing Surfaces

The thermophysical properties of the enclosing surfaces are described by specifying the thermal conductivity, specific heat, emissivity, density, and thickness of the enclosing surfaces for each room. If the thermophysical properties of the enclosing surfaces are not included, FAST will treat them as adiabatic (no heat transfer). Since most of the heat conduction is through the ceiling and since the conduction calculation takes a significant fraction of the computation time, it is recommended that initial calculations be made using the ceiling only. Adding the walls generally has a small effect on the results and the floor contribution is usually negligible. Clearly, there are cases where the above generalization does not hold, but it may prove to be a useful screening technique. Currently, thermal properties for materials are read from a thermal database file unique to FAST. The data in the file for FAST simply gives a name (such as CONCRETE) which is a pointer to the properties in the thermal database. (For computers which do not support extensions, the ".DAT" is dropped.) For the PC version, this is an installation parameter. All of these specifications are optional. The thermal properties are assumed to be constant; that is, we do not account for the variation with temperature or water content.

The thermophysical properties are specified at one condition of temperature, humidity, *etc.* There can be as many as three layers per boundary, but they are specified in the thermal database itself.

Label	Parameter	Comments	Units
CEILI	(N)	The label CEILI indicates that the names of thermophysical properties on this line describe the ceiling material. If this parameter is present, there must be an entry for each compartment.	
WALLS	(N)	The label WALLS indicates that the names of thermophysical properties on this line describe the wall material. If this parameter is present, there must be an entry for each compartment.	
FLOOR	(N)	The label FLOOR indicates that the names of thermophysical properties on this line describe the floor material. If this parameter is present, there must be an entry for each compartment.	

Examples:

```
CEILI OFF         REDOAK     CONCRETE
WALLS CONCRETE    CONCRETE   CONCRETE
```

The corresponding thermal data base might appear as

```
CONCRETE    1.75  1000.   2200.   0.1500  0.94
BRICK       0.18   900.    790.   0.016   0.90
REDOAK      0.15  1300.    640.   0.025   0.99
```

The names of the materials can be **any** ASCII string up to 8 characters. So a valid name is $%#@**% although this admittedly does not convey much information. The key word "OFF" is used to tell the model not to compute the heat loss for the ceiling in compartment (1). In this case the FLOOR parameter is not present at all, so there will be no heat transfer through the floor in any room and the calculation will not be done for the ceiling in compartment (1), where the key word "off" is present. This is most useful for doing the heat transfer calculation in the burn room and adjacent rooms and then turning it off in distant compartments.

A.7 Fire Specifications

The fire specifications allow the user to describe the fire source in the simulation. The location and position of the fire is specified along with the chemical properties of the fuel. Finally, the fire is described with a series of mass loss rate, fuel height, and fuel area

inputs. All of these specifications are optional and each line requires a single number. The defaults for the fire specification is a methane burner in the center of compartment (1). The defaults shown for each key word reflect the values for methane.

Label	Parameter	Comments	Units
LFBO	**Room of Fire Origin**	Room of fire origin is the room number in which the fire originates. Default is 1.	
LFBT	**Fire Type**	This is a number indicating the type of fire. 1 Unconstrained fire 2 Constrained fire. The default is 1. See sections 4.5 and 5.5 for a discussion of the implications of this choice.	
LFPOS	**Fire Position**	The fire position is the area of the room in which the fire originates and is one of the following values: 1 Center of the room, 2 Corner of the room, or 3 Along a wall of the room, but not near a corner of the room. The fire position is used to account for the entrainment rate of the plume, which depends on the location of the fire plume within the compartment. Fire positions 2 and 3 should only be used when the fire is very close to the corner or wall respectively. The default is 1.	
CHEMI	(6)	Chemical kinetics and miscellaneous parameters.	
	Molar Weight	Molecular weight of the fuel vapor. This is the conversion factor from mass density to molecular density for "tuhc." Default is 16. It is used only for conversion to ppm, and has no effect on the model itself.	

	Relative Humidity	The initial relative humidity in the system. This is converted to kilograms of water per cubic meter from the table from "Dynamical and Physical Meteorology" by Haltiner and Martin (1957)	%
	Limiting Oxygen Index	The limit on the ratio of oxygen to other gases in the system below which a flame will not burn. This is applicable only to type (LFBT) 2 or later fires. The default is 10.	%
	Heat of Combustion	Heat of combustion of the fuel. Default is 50000000.	J/kg
	Initial Fuel Temperature	Typically, the initial fuel temperature is the same as the ambient temperature as specified in the ambient conditions section.	K
	Gaseous Ignition Temperature	Minimum temperature for ignition of the fuel as it flows from a compartment through a vent into another compartment. The default is the **initial fuel temperature**.	K
LFMAX	**Number of Intervals**	This is the number of time intervals for the mass loss rate, fuel height and species inputs. The mass loss rate, fuel height and species are entered as series of points with respect to time. This is referred to in this document as a specified fire. A sufficient number of intervals should be selected to provide a reasonable approximation (using straight line segments) for the input variables which specify the fire. A example of this is shown in figure 8. The mass loss rates P_1-P_7 are specified over the time intervals I_1-I_6. The number of points specified must be one greater than the number of time intervals. For example, if there are six mass loss points there should be a total of five time intervals (or one interval between every two consecutive points). The maximum number of intervals allowed in version 18 of FAST is 21.	
FTIME	**Time Interval (LFMAX)**	Time interval is the time between each point (mass loss rate, fuel height and species) specified for the fire. The total duration of the fire is the	s

		sum of the time intervals. This time is independent of the simulation time which is specified for the TIMES label. If the simulation time is longer than the total duration of the fire, the final values specified for the fire (mass loss rate, fuel height, fuel area, and species) will be continued until the end of the simulation. The number of values on the line must equal the number of time intervals specified by LFMAX, above.	
FMASS	**Mass Loss Rate (LFMAX+1)**	The rate at which fuel is pyrolyzed at times corresponding to each point of the specified fire.	kg/s
FHIGH	**Fuel Height (LFMAX+1)**	The height of the base of the flames above the floor of the room of fire origin for each point of the specified fire.	m
FQDOT	**Heat Release Rate (LFMAX+1)**	The heat release rate of the specified fire.	W

With the three parameters, the heat of combustion (HOC) from CHEMI, FMASS and FQDOT, the pyrolysis and heat release rate are over specified. The model uses the last two of the three to obtain the third parameter. That is, if the three were specified in the order HOC, FMASS and FQDOT, then FQDOT would be divided by FMASS to obtain the HOC for each time interval. If the order were FMASS, FQDOT and HOC, then the pyrolysis rate would be determined by dividing the heat release rate by the heat of combustion. If only two of the three are given, then those two will determine the third, and finally, if none or only one of the parameters is present, the defaults shown will be used.

Example:

```
LFBO     1
LFBT     1
LFPOS    1
CHEMI    0.0  0.0 10.  18100000. 300.
LFMAX    7
FMASS  .014  .0014 .025  .045  .050  .0153 .0068 .0041
FAREA  .5    .5    .5    .5    .5    .5    .5    .5
FHIGH  .25   .25   .25   .25   .25   .25   .25   .25
FTIME      20.   20.   50.   50.  100.  100.  400.
```

In the example, a specified fire (LFBT 1) originates in room number 1 (LFBO 1) in the center of the room (LFPOS 1). A seven segment (LFMAX) fire is specified. The fuel burns

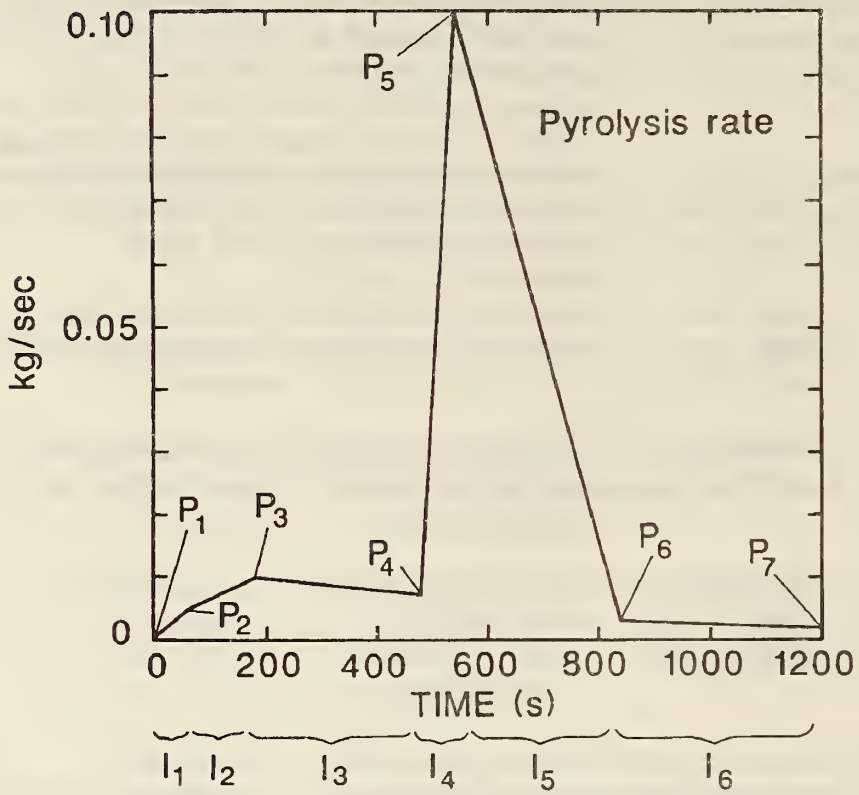

Figure 8 Pyrolysis rate for LFMAX=6.

with a heat of combustion of 18100000 J/kg. The initial relative humidity is 0%, the molecular weight is 16 (zero is not allowed, so the default is used) and the limiting oxygen index is 10%. Since the type of fire is 1, an unconstrained fire, this latter parameter has no meaning in this context.

```
LFBT     2
LFMAX    7
FMASS  .014   .0014 .025   .045   .050   .0153 .0068 .0041
FAREA  .5     .5    .5     .5     .5     .5    .5    .5
FHIGH  .25    .25   .25    .25    .25    .25   .25   .25
FTIME       20.   20.   50.   50.  100.  100.  400.
```

In this example, the specified fire is constrained with a limiting oxygen index of 1%. Since LFBO is not given, the default compartment (1) is used, and the position of the fire is in the center of the room. The default heat of combustion of 50000000 kJ/kg is used.

A.8 Species Production

Species production rates are specified in the manner similar to the fire, entering the rates as a series of points with respect to time. The species which are followed by FAST are

- Carbon Dioxide
- Carbon Monoxide
- Concentration-Time Product
- Hydrogen Cyanide
- Hydrogen Chloride
- Nitrogen
- Oxygen
- Soot (Smoke Density)
- Total Unburned Hydrocarbons
- Water

For a type one (LFBT=1) fire, only the concentration-time product of pyrolysate(ct), hydrogen cyanide(hcn) and hydrogen chloride(hcl) can be specified. No other species are followed. For a type two (LFBT=2) fire, nitrogen, oxygen, carbon dioxide, carbon monoxide, soot, unburned fuel and water are followed. In all cases, the unit of the production rates is kg/kg. However, the meaning of the production rates is different for the several types of species. For either fire, the production rates for ct, hcn and hcl are with respect to the pyrolysis rate of the fuel. For the others, carbon monoxide, water, *etc.*, the production rate is specified with respect to the basic carbon production in the form of a ratio with carbon dioxide. For carbon monoxide, for example, the specification will be CO/CO_2. Thus we can not consider a pure hydrogen flame, but this is unlikely in the situations of interest.

Label	Parameter	Comments	Units
SPECIES	**(LFMAX+1)**	For each species desired a series of production rates are specified for each of the time points input for the specified fire. The program performs a linear interpolation between these points to determine the time of interest.	
HCN, HCL and CT	**Production Rate of the Fuel**	Units are kilogram of species produced per kilogram of fuel burned. The input for CT is the kilograms of "toxic" combustion products produced per kilogram of fuel burned.	kg/kg

HCR	**Production Rate of the Fuel**	The mass ratio of hydrogen to carbon *as it becomes available from the fuel*. This parameter affects primarily the rate of production of water.	kg/kg
O2	**Production Rate of the Fuel**	The mass ratio of oxygen to carbon *as it becomes available from the fuel.*	kg/kg
OD	**Yield**	The ratio of the mass of carbon to carbon dioxide produced by the oxidation of the fuel.	kg/kg
CO	**Yield**	The ratio of the mass of carbon monoxide to carbon dioxide produced by the oxidation of the fuel.	kg/kg

A.9 Files

There are several files which FAST uses to communicate with its environment. They are 1) a configuration file, 2) the thermal database, 3) a "history" file, and 4) a restart file. The output of the simulation may be written to a disk file for further processing by programs such as FASTplot or to restart FAST. At each interval of time as specified by the history interval in the TIMES label, the output is written to the file specified. For efficient disk storage and optimum speed, the data is stored in an internal format and cannot be read directly with a text editor.

Label	Parameter	Comments	Units
DUMPR	**Dump File**	The name specifies a file (up to 17 characters) to which the program outputs for plotting are written. **Dump file** is an optional input. If omitted, the file will not be generated. Note that in order to obtain a history of the variables, this parameter must be specified and also the historyer interval (under TIMES) must be set to a non-zero value.	

RESTR	**Restart File**	The name specifies a file (up to 17 characters) from which the program reads data to restart the model. This data must have been generated (written) previously with the dump parameter discussed earlier. A time step is given after the name of the file and specifies at what time the restart should occur.
THRMF	**Thermal Database**	The name specifies a file (up to 20 characters) from which the program reads thermophysical data. If this parameter is not specified, then either the default (THERMAL.DAT) is used, for the name is read from the configuration file.
DEFCG	**Configuration File**	The name specifies a file (up to 20 characters) from which the program reads configuration information data.

Example:

```
DUMPR FAST1.DAT
RESTR filename n
THRMF thermal.tpf
```

where "filename" was created in a previous run using the DUMP parameter. "n" specifies the starting time and must be one of the times at which a history was generated. As an example, if a data set were run with

```
VERSN   18 title...
TIMES   360 60 10 0 0
.
.
DUMPR   MYFILE
```

then every 10 s a snap shot of the time histories of all variables would be generated. So a restart might be done at 300 seconds with the following

```
VERSN   18 new title
TIMES   900 60 0 0 0
.
.
RESTRT  MYFILE 300
```

with no requirement that the restart must be at the last history point. The only caveat is to check the listings to be sure that a history was generated at the desired point. For those cases where too many history intervals are requested, the interval is recalculated, and a message is written to the output device.

A.10 Graphics Specification

A graphics specification can be added to the data file. Details of the meaning of some of the parameters is best left to the discussion of the device independent graphics software used by FAST [2]. However, the information necessary to use it is straightforward. The general structure is similar to that used for the building and fire specification. One must tell the program "what to plot," "how it should appear," and "where to put it."

The key words for "where to put it" are

DEVICE	where to plot it
BAR	bar charts
GRAPH	specify an x-y plot
TABLE	put the data into a table
PALETTE	specify the legend for CAD views
VIEW	show a perspective picture of the structure
WINDOW	the size of the window in "user" space.

The complete key word is required. That is, for the "where to put it" terms, no abbreviations are allowed. Then one must specify the variables to be plotted. They are

VENT, HEAT, PRESSUR, WALL, TEMPERA, INTERFA,
H_2O, CO_2, CO, OD, O_2, TUHC, HCN, HCL, CT

As might be expected, these are the similar key words to those used in the plotting program, FASTplot. In this case, we have a reduced set. The application and use of FAST and FASTplot are different.

For each key word there are parameters to specify the location of the graph, the colors and finally titles as appropriate. For the variables, there is a corresponding pointer to the graph of interest.

The form of each "where to put it" variable is described below

Label	Parameter	Comments
DEVICE	**Plotting Device**	The **Plotting Device** specifies the hardware device where the graphics is to be displayed. For the PC version, this key word should be omitted. If it must be included for compatibility reasons, set it to 4. For other computers, it is installation dependent. In general it specifies which device will receive the output.

WINDOW	(6)	The window label specifies the **user space** for placement of graphs, views,...
	Xl	left hand side of the graph in any user desired units.
	Yb	bottom of the graph in any user desired units.
	Zf	forward edge of the 3D block in any user desired units.
	Xr	right hand side of the graph in any user desired units.
	Yt	top of the graph in any user desired units.
	Zb	rear edge of the 3D block in any user desired units. These definitions refer to the 3D plotting block that can be seen. The most common values (which are also the default) are

Xl = 0.
Yb = 0.
Zf = 0.
Xr = 1279.
Yt = 1023.
Zb = 10.

This is not a required parameter; however, it is often convenient to define graphs in terms of the units that are used. For example, if one wished to display a house in terms of a blueprint, the more natural units might be feet. In that case, the parameters might have the values

Xl = 0.
Yb = 0.
Zf = 0.
Xr = 50.
Yt = 25.
Zb = 30.

GRAPH	(10)	Up to five graphs may be displayed at one time on the graphics display. Each graph is identified by a unique number (1-5) and placed in the window at a specified location. Xl,Yb,Zf,Xr,Yt and Zb have a meaning similar to WINDOW. However, here they specify where in the window to put the graph.
	Graph Number	The number to identify the graph. Allowable values are from 1 to 5. The graphs must be numbered consecutively, although they do not have to be given in order. It is acceptable to define graph 4 before graph 2 but if graph 4 is to be used, then graphs 1 through 3 must also be defined.
	Xl	Left hand side of the graph within the window in the same units as that of the window.
	Yb	Bottom of the graph within the window in the same units as that of the window.
	Zf	Forward edge of the 3D (three dimensional) block within the window in the same units as that of the window.
	Xr	Right hand side of the graph within the window in the same units as that of the window.
	Yt	Top of the graph within the window in the same units as that of the window.
	Zb	Back edge of the 3D block within the window in the same units as that of the window.
	Color	The color of the graph and labels which is specified as an integer from 1 to 15. Refer to DEVICE (NBSIR 85-3235) for the colors corresponding to the values for the color.
	Abscissa Title	Title for the abscissa (horizontal axis). To have blanks in the title, use the underscore character "_".

	Ordinate Title	Title for the ordinate (vertical axis). To have blanks in the title, use the underscore character "_".
TABLE	(7)	Up to five tables may be displayed at one time on the graphics display. Each table is identified by a unique number and placed in the window at a specified location. Xl,Yb,Zf,Xr,Yt and Zb have a meaning similar to WINDOW. However, here they specify where in the window to put the table.
	Table Number	The table number is the number to identify the table. Allowable values are from 1 to 5. The tables must be numbered consecutively, although they do not have to be given in order. It is acceptable to define table 4 before table 2 but if table 4 is to be used, then tables 1 through 3 must also be defined.
	Xl	Left hand side of the table within the window in the same units as that of the window.
	Yb	Bottom of the table within the window in the same units as that of the window.
	Zf	Forward edge of the 3D block within the window in the same units as that of the window.
	Xr	Right hand side of the table within the window in the same units as that of the window.
	Yt	Top of the table within the window in the same units as that of the window.
	Zb	Back edge of the 3D block within the window in the same units as that of the window.

VIEW	(24)	Up to five views may be displayed at one time on the graphics display. Each view is identified by a unique number and placed in the window at a specified location. Xl,Yb,Zf,Xr,Yt and Zb have a meaning similar to WINDOW. However, here they specify where in the window to put the view.
	View Number	View number is the number to identify the view. Allowable values are from 1 to 5. The views must be numbered consecutively, although they do not have to be given in order. It is acceptable to define view 4 before view 2 but if view 4 is to be used, then views 1 through 3 must also be defined.
	Xl	Left hand side of the view within the window in the same units as that of the window.
	Yb	Bottom of the view within the window in the same units as that of the window.
	Zf	Forward edge of the 3D block within the window in the same units as that of the window.
	Xr	Right hand side of the view within the window in the same units as that of the window.
	Yt	Top of the view within the window in the same units as that of the window.
	Zb	Back edge of the 3D block within the window in the same units as that of the window.
	File	File is the filename of a compatible "BUILD" file, as discussed later.
	Transform Matrix	The **Transform Matrix** is a 16 number matrix which allows dynamic positioning of the view within the window. The matrix (1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1) would show the image as it would appear in a display from BUILD.

PALETTE	(15)	The PALETTE label performs a specialized function for showing colors on the views. A four entry table is created and used for each type of filling polygon used in a view. Up to five palettes may be defined. Each palette is identified by a unique number and placed in the window at a specified location. Xl, Yb, Zf, Xr, Yt and Zb have a meaning similar to WINDOW. However, here they specify where in the window to put the palette.
	Palette Number	Palette number is the number to identify the palette. Allowable values are from 1 to 5.
	Xl	Left hand side of the palette within the window in the same units as that of the window.
	Yb	Bottom of the palette within the window in the same units as that of the window.
	Zf	Forward edge of the 3D block within the window in the same units as that of the window.
	Xr	Right hand side of the palette within the window in the same units as that of the window.
	Yt	Top of the palette within the window in the same units as that of the window.
	Zb	Back edge of the 3D block within the window in the same units as that of the window.
	Color and Label	There are four pairs of color/text combinations, each corresponding to an entry in the palette. The color number is an integer from 1 to 15 and the text can be up to 50 characters (but remember the 128 character maximum). As before, spaces are indicated with an underscore character "_".
BAR	(9)	Up to five bar charts may be displayed at one time on the graphics display. Each bar chart is identified by a unique number and placed in the

		window at a specified location. Xl,Yb,Zf,Xr,Yt and Zb have a meaning similar to WINDOW. However, here they specify where in the window to put the bar chart.
	Bar Chart Number	The number to identify the bar chart. Allowable values are from 1 to 5.
	Xl	Left hand side of the bar chart within the window in the same units as that of the window.
	Yb	Bottom of the bar chart within the window in the same units as that of the window.
	Zf	Forward edge of the 3D block within the window in the same units as that of the window.
	Xr	Right hand side of the bar chart within the window in the same units as that of the window.
	Yt	Top of the bar chart within the window in the same units as that of the window.
	Zb	Back edge of the 3D block within the window in the same units as that of the window.
	Abscissa Title	Title for the abscissa (horizontal axis). To have blanks in the title, use the underscore character "_".
	Ordinate Title	Title for the ordinate (vertical axis). To have blanks in the title, use the underscore character "_".
LABEL	(10)	Up to five labels may be displayed at one time on the graphics display. Each label is identified by a unique number and placed in the window at a specified location. Xl, Yb, Zf, Xr, Yt, and Zb have a meaning similar to WINDOW. However, here they specify where in the window to put the label. It is assumed that time is always to be displayed if any labels are present. To this end, label 1 is always used for the time in the units HH:MM:SS.

Label Number	Label number is the number to identify the label. Allowable values are from 1 to 5.
Xl	Left hand side of the label within the window in the same units as that of the window.
Yb	Bottom of the label within the window in the same units as that of the window.
Zf	Forward edge of the 3D block within the window in the same units as that of the window.
Xr	Right hand side of the label within the window in the same units as that of the window.
Yt	Top of the label within the window in the same units as that of the window.
Zb	Back edge of the 3D block within the window in the same units as that of the window.
Text	The text to be displayed within the label. To have blanks in the title, use the underscore character "_".
Angle1, Angle2	Angles for display of the label in a right cylindrical coordinate space. At present only the first angle is used and represents a positive counterclockwise rotation; set the second angle to zero. Both angles are in radians.

In order to see the variables, they must be assigned to one of the above displays. This is accomplished with the variable pointers as

```
(Variable) (nmopq) (Compartment) (Layer).
            12345
```

Variable is one of the available variables VENT, HEAT, PRESSUR, WALL, TEMPERA, INTERFA, N_2, O_2, CO_2, CO, HCN, HCL, TUHC, H_2O, OD, CT used as a label for the line. The species listed correspond to the variable "SPECIES" in FASTplot. In the variable list of FAST, all are contained in the variable TOXICT. **(nmopqr)** is a vector which points to

```
index   display in
(1) n -> bar chart
(2) m -> table
(3) o -> view
(4) p -> label
(5) q -> graph
```

respectively. These numbers vary from 1 to 5 and correspond to the value of "n" in the "where to put it" specification. **Compartment** is the compartment number of the variable and **Layer** is "U" or "L" for upper and lower layer, respectively.

Examples:

```
WINDOW      0   0 -100  1280  1024   1100
GRAPH 1   250. 170.   0. 1220. 900.   10.  5 TIME  CELSIUS
LABEL 1   970. 960.   0. 1231. 1005.  10. 15 00:00:00                  0.  0.
LABEL 2   690. 960.   0.  987. 1005.  10. 13 TIME_[S]                  0.  0.
LABEL 3   90.  920.   0.  730. 1020.  10. 4 Single_Compartment_demo   0.  0.
LABEL 4   400. 610.   0.  687. 660.   10. 1 U_layer_temperature      .1   0.
LABEL 5   400. 270.   0.  687. 320.   10. 1 l_layer_temperature      .0   0.
TEMPERA 0 0 0 0 1   1 U
TEMPERA 0 0 0 0 1   1 L
```

In this case, a new window is defined, along with one graph and five labels. Both temperature variables are assigned to graph 1. One quirk is not obvious. It is assumed that time is always to be displayed if any labels are present. To this end, label 1 is always used for the time in the units HH:MM:SS. Graph 1 has the label "TIME" on the abscissa and "CELSIUS" on the ordinate.

```
WINDOW      0    0  -100  1280  1024   1100
GRAPH   1  120.  300.    0.  600.  920.   10.  3 TIME  PPM
GRAPH   2  740.  300.    0.  1220. 920.   10.  3 TIME  CELSIUS
LABEL   1  970.  960.    0. 1231. 1005.   10. 15 00:00:00        0.      0.
LABEL   2  690.  960.    0.  987. 1005.   10. 13 TIME_[S]        0.      0.
LABEL   3  200.  050.    0.  520.  125.   10.   14 CO¦D2¦O_CONCENTRATION 0. 0.
CO      0 0 0 0 1   1 U
TEMPERA 0 0 0 0 2   1 U
TEMPERA 0 0 0 0 2   1 L
```

This file sets up two graphs with the CO data from the upper layer of compartment (1) in the first graph and both the upper and lower layer temperatures displayed on the second graph.

```
WINDOW      0    0  -100  1280  1024   1100
GRAPH   1  150.  300.    0.  620.  920.   10.  3 TIME  PPM
LABEL   1  390.  960.    0.  651. 1005.   10. 15 00:00:00        0.      0.
LABEL   2  110.  960.    0.  407. 1005.   10. 13 TIME_[S]        0.      0.
LABEL   3  200.  050.    0.  520.  125.   10.   14 O¦D2¦O_CONCENTRATION 0. 0.
TABLE   1  700.  300.    0. 1200.  920.   10.
HEAT    0 1 0 0 0   1 U
O2      0 1 0 0 1   1 U
TEMPERA 0 1 0 0 0   1 U
TEMPERA 0 1 0 0 0   1 L
```

Here the four variables HEAT, O2, and TEMPERATURE are displayed in table 1 and O2 is shown in graph 1.

```
WINDOW       0     0  -100  1280  1024   1100
VIEW    1  800.  390.  150. 1200.  900.  200. DEMOFA.DAT 1.41 .48 1.33 0....
VIEW  2  420.  200.   50.  720.  500.  100. DEMOFA.DAT 1.53 -.46 1.21 0. ...
GRAPH   1   50.  290.    0.  300.  490.   10. 13 TIME  PPM
GRAPH   2  150.  650.    0.  500.  850.   10. 13 TIME  m¦U-1
GRAPH   3  510.  690.    0.  740.  890.   10. 13 TIME  CELSIUS
GRAPH   4  810.  120.    0. 1160.  320.   10. 13 TIME  HEIGHT
LABEL   1  760.  960.    0. 1021. 1005.   10. 14 00:00:00        0.      0.
LABEL   2   50.   10.    0.  250.   70.   10.  2 TEST_IN_ONE_ROOM   0.57079 0.
LABEL   3   70.  960.    0.  367. 1005.   10. 13 FIRE_[W]        0.      0.
LABEL   4  480.  960.    0.  777. 1005.   10. 13 TIME_[S]        0.      0.
LABEL   5  300.  960.    0.  475. 1005.   10. 14 _______         0.      0.
TABLE   1  220.   50.    0.  520.  250.   10.
HEAT    0 0 0 5 0   1 U
OD      0 1 1 0 0   1 U
OD      0 0 2 0 2   1 U
CO      0 1 1 0 1   1 U
CO      0 0 2 0 0   1 U
TEMPERA 0 1 1 0 3   1 U
TEMPERA 0 0 2 0 0   1 U
INTERFA 0 1 1 0 4   1 U
INTERFA 0 0 2 0 0   1 U
```

Two views are specified, both emanating from the file "demofa.dat" with different transforms. Four graphs, three labels and one table will be displayed. All variables will be taken from the upper layer in compartment (1), and they will go to both views, in determining the hazard calculation. The variables will also be shown in table 1 and in the four graphs, respectively.

A.11 Mechanical Ventilation

Label	Parameter	Comments
MVOPN	(5)	Connect a compartment to a node in the mechanical ventilation system.
	Compartment Number	Specify the compartment number
	Duct Work Node Number	Corresponding node in the mechanical ventilation system to which the compartment is to be connected.
	Orientation	V for vertical or H for horizontal

	Height	Height of the duct opening above the floor of the compartment.	
	Area	Area of the opening into the compartment	
MVDCT	(9)	Specify a piece of duct work.	
	Node Number	First node number. This is a node in the mechanical ventilation scheme, not a compartment number (see **MVOPN**).	
	Node Number	Second node number.	
	Length	Length of the duct	m
	Diameter	All duct work is assumed to be circular. Other shapes must be approximated by changing the flow coefficient. This is done implicitly by network models of mechanical ventilation and forced flow, but must be done explicitly here.	m
	Absolute Roughness	Roughness of the duct.	m
	Flow Coefficient	Flow coefficient to allow for an expansion or contraction at the end of the duct which is connected to node number one. To use a straight through connection (no expansion or contraction) set to zero.	
	Area	Area if the expanded joint.	m^2
	Flow Coefficient	Coefficient for second node.	
	Area	Area at the second node.	m^2
MVFAN	(9)	Specify fan curve with power law coefficients. There must be at least one coefficient.	
	First Node	First node in the mechanical ventilation system to which the fan is connected.	
	Second Node	Second node to which the fan is connected.	

	Minimum Pressure	Lowest pressure of the fan curve. Below this values, the flow is assumed to be constant.	Pa
	Maximum Pressure	Highest pressure at which the fan will operate. Above this point, the flow is assumed to stop.	Pa
	Coefficients	At least one, and a maximum of five coefficients to specify the flow as a function of pressure.	
INELV	(2×n)	Specify **interior** elevations of the mechanical ventilation nodes. All nodes can be specified, but the exterior nodes, that is those connected to a compartment, will be set to the compartment elevation. The heights are absolute heights above the reference datum specified by TAMB. The heights are specified in pairs, the node number followed by the height.	(#,m)

A.12 Miscellaneous

CNVG	(3)	Diagnostic parameters	
	Approximate Conduction	This is a logical switch. If set to zero, the usual heat transfer through partitions is calculated. If set to any non zero value then the semi-infinite slab approximation is used.	0
	Convergence Factor	Modifies the convergence criterion used by the solver for the relative error at each time step.	1.0
	Pressure Damping	Used to damp pressure fluctuations.	1.0

The diagnostic parameters must be used with extreme caution. These parameters can not be set in the data editor for the main model. While running computations within the data editor, the speed enhancement for the conduction can be utilized. Normally, a linear parabolic equation is solved for the heat transfer through each partition. The solution is *via* a successive over-relaxation technique. When the walls are thick, or the simulation time will be very short it is reasonable to assume that the thermal wave will not reach the outside boundary. In this case, there is an analytic solution to the conduction equation which can be written in terms of the "error function," ERFC [9]. The other two parameters modify the

convergence criteria used by the numerical solver. The default values are unity, and should normally be left at those values.

```
Example 1:

MVDCT  1 3 20. .15 .19E-3  3.30  .01767  0. 0.
MVDCT  3 4 44. .2  .19E-3  0.0   0.0     0. 0.
MVFAN   4 5   0.0 140.   0.140 3.170E-04 -1.803E-05  1.898E-07 -8.104E-10
MVOPN  1 1 H 0. 1.
MVOPN  4 9 V 0. 2.5
INELV  3  0   4 0  5 0  7 0 8 0

Example 2:

MVDCT  1 3 20. .15 .19E-3  3.30  .01767  0. 0.
MVDCT  2 3 20. .15 .19E-3  2.94  .01767  0. 0.
MVDCT  3 4 44. .2  .19E-3  0.0   0.0     0. 0.
MVDCT  5 6 40. .2  .19E-3  .51   .03142  0. 0.
MVDCT  7 4 40. .2  .19E-3  .51   .03142  0. 0.
MVDCT  9 8 40. .2  .19E-3  .51   .03142  0. 0.
MVFAN  4 5  0.0 140.   0.140 3.170E-04 -1.803E-05  1.898E-07 -8.104E-10
MVFAN  8 7  0.0 140.   0.140 3.170E-04 -1.803E-05  1.898E-07 -8.104E-10
MVOPN  1 1 H 0. 1.
MVOPN  2 2 H 0. 1.5
MVOPN  3 6 H 0. 2.5
MVOPN  4 9 H 0. 2.5
INELV  3  0   4 0  5 0  7 0 8 0
```

This describes a network which is shown in figure 9.

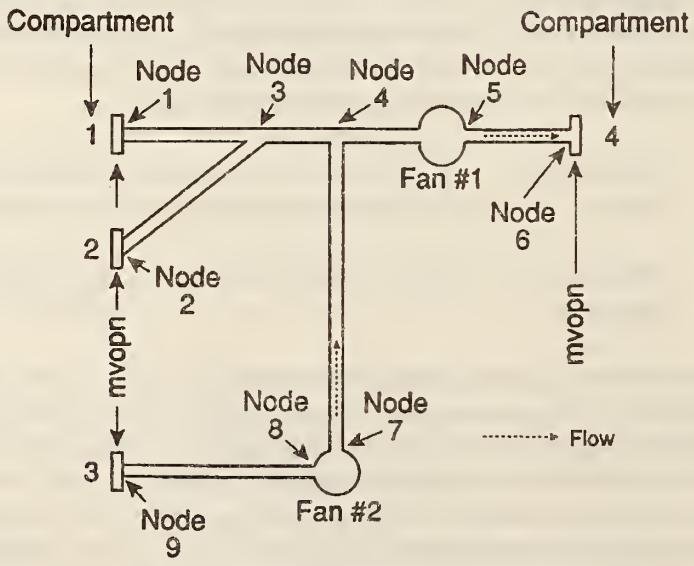

Figure 9. Node arrangement for example 2.

Appendix B: LABELLED COMMON BLOCKS DOCUMENTATION

There are several common blocks in the suite of programs. They are used to pass various kinds of information, and the data is grouped according to the use of the information. The first set is the main data, which is the working data set. This data is saved by the history subroutine "DUMPER." The second is the shell information which contains information about the computer envirnoment. The third is used for thermophysical and other properties. Finally, there is an unlabelled common block which contains the current environment. This latter is set by the data copy routine.

The "logfiles" are specified by an "nv" extension, the base editing file by the "ins" extension, and the actual files used during a compilation by the "inc" extension.

The name of the "include" files and the corresponding common blocks are

1. CPARAMS.INC	parameter statements, *i.e.* nr=7,...
2. CFAST.INC	main common block (2) moco1a and moco1b
3. PARAMS.INC	intermodule communication
4. THERMP.INC	physical properties such as conduction
5. CENVIRO.INC	environment - set by the data copy routines
6. CSHELL.INC	shell variables such as the screen colors
7. PRECIS.INC	specify the precision of the floating point data
8. CFIO.INC	input/output buffers for the data editor

Note that the actual files are "ins" files. They are converted to "inc" files by the filt command. This is an important step. The "inc" should never be edited directly. Several of the common blocks require initialization. This is done with BLOCK DATA sub probprograms. They are listed at the end of the PROGRAM GLOSSARY.

B.1 Parameter header file, CPARAMS.INC

MARG	maximum number of file name (arguments) on the command line
MAXFIL	maximum number of history files that can be accessed in a session
MBR	maximum number of branches
MCON	maximum number of connections to a node
MDT	maximum number of ducts
MEXT	maximum number of exterior nodes
MFAN	maximum number of fans
MFCOE	maximum number of fan coefficients
MFT	maximum number of simple fittings
MNODE	maximum number of nodes

MOPT	maximum number of options allowed on the command line
MXSLB	maximum number of different materials in a wall
NOPTN	number of options available in the plotting routine
NR	maximum number of compartments
NN	maximum number of nodes in a boundary (walls, ceilings and floor)
NT	maximum number of equations to be solved
NTHMX	maximum number of thermal definitions
NV	maximum number of time intervals
NVAL	maximum number of *data points* in a list of data
NVAR	maximum number of different *variables* that can be plotted
NVM	maximum number of curves that can be plotted simultaneous
NS	maximum number of species to be tracked
NWAL	number of discrete wall surfaces (ceiling, upper wall ...)

UPPER,LOWER upper, lower layer pointers (=1,2)

and the current values

MARG = 3
MAXFIL = 10
MBR = MFAN+MDT
MCON = 3
MDT = 2*NR+2
MEXT = 2*NR
MFAN = 5
MFCOE = 5
MFT = 2*MDT
MNODE = 4*MDT
MOPT = 5
MXSLB = 3
NN = 36
NOPTN = 20
NR = 7
NS = 10
NT = 4*NR(MAIN EQU) + 2*NR*NS(SPECIES) * 4*NR(HCL) + 4*NR(SMOKE)
(or NT = 12*NR+2*NR*NS)
NTHMX = 56
NV = 21
NVAL = 250
NVAR = 21
NVM = 20
NWAL = 4

B.2 Definitions of the variables CFAST.INC

Variables in the common block moco1a (CFAST.INC)

AA(NR,NR,4)	flow from lower layer to lower layer (kg/s)
ACTIVS(NS)	logical switch to tell which species are active (interacts with "allowed")
AFIRED(NV)	area of fire (m^2)
AO(IFT)	area of simple fitting ift (m^2)
APS(NR)	current area of the specified fire (m^2)
AR(NR)	floor area of a compartment (in current version ceiling=floor)
AS(NR,NR,4)	flow from lower to upper layer (kg/s)
ASL(NR,NR,4)	entrainment from upper into lower layer (kg/s)
BFIRED(NV)	burning rate (kg/s)
BFLO(IB)	mass flow rate through branch ib (kg/s)
BR(NR)	breadth of a compartment (m)
BW(NR,NR,4)	width of vent (m) (modified by qcvent)
CCO2(NV)	net carbon production rate (fraction relative to co2)
CE(IB)	effective mass flow coefficient for resistance branch ib
CO(IFT)	flow coefficient for simple fitting ift
COCO2(NV)	relative co/co2 production rate
CP	heat capacity of AIR at constant pressure (J/kg/K)
CRDATE(3)	creation date of the model (day, month and year)
CW(MXSLB,NWAL,NR)	specific heat of a thermal material
DA(ID)	area of duct id (m^2)
DE(ID)	effective diameter of duct id (m)
DELTAT	time step used by the model, currently 1.0 seconds
DFMAX(K)	derivative of fan curve at hmax(k)
DFMIN(K)	derivative of fan curve at hmin(k)
DL(ID)	length of duct id (m)
DPZ(I,K)	hydrostatic pressure difference between node i and kth node
DR(NR)	depth of a compartment (m)
DUCTAR(ID)	absolute roughness of duct walls
EME(NR)	plume entrainment rate (kg/s)
EMP(NR)	pyrolysis rate of the fire source (kg/s)
EMS(NR)	plume flow rate into the upper layer (kg/s)
EPW(NWAL,NR)	emissivity of the interior wall surface (non)
FKW(MXSLB,NWAL,NR)	thermal conductivity of a material slab
FLW(MXSLB,NWAL,NR)	thickness of a slab (m)
G	gravitational constant (9.806 m/s)
GAMMA	c_p/c_v for air - 1.4
GMWF	gram molecular weight (in grams, default ->16)
HCOMBA	heat of combustion - initialization only
HCRATIO(NV)	hydrogen/carbon ratio in the fuel - time dependent
HEATLP(NR)	heat release rate in the plume in the lower layer (W)

HEATUP(NR)	heat release rate in the plume in the upper layer (W)
HEATVF(NR)	heat release in a vent (sum of all vents between compartments)
HFIRED(NV)	height of the base of the fire - time dependent (m)
HFLR(NR)	absolute height of the floor of a compartment (m)
HH(NR,NR,4)	top of vent (soffit) - distance from floor (m)
HHP(NR,NR,4)	absolute height of the soffit (m)
HL(NR,NR,4)	height of the sill relative to the floor (m)
HLP(NR,NR,4)	absolute height of the sill (m)
HMAX(K),HMIN(K)	max and min head pressure for fan(Pa)
HOCBMB(NV)	heat of combustion of a specified fire (J/kg)
HR(NR)	interior height of a compartment (m)
HRL(NR)	absolute height of the floor (m)
HRP(NR)	absolute height of the ceiling (m)
HVBCO(K,J)	coefficients of fan curve polynomial
HVELXT(II)	elevation of exterior nodes relative to station (m)
HVEXCN(MEXT,NS)	species concentration at external nodes (kg/m^3)
HVFLOW(I,J)	mass flow rate to node i from the jth node to which it is connected
HVGHT(I)	elevation of node i
HVNODE(I,J)	mapping between external and internal nodes (2,MNODE)
HVP(I)	relative pressure at node i
HWJ(NW,NR)	hcl density on the wall ($grams/m^2$, initialized to 0)
IBRD(ID)	pointer to resistive branch with duct id
IBRF(IFT)	pointer to resistive branch with fitting ift
IC(I,K)	pointer to kth resistive branch connected to node i
IDIAG	not used
IFIRED	current interpolation time for specified fire - integer pointer
IN(I,K)	pointer to kth node connected to node i in hvac system
ITMMAX	maximum number of time steps (#)
ITMSTP	current time step (#)
IVERS	current version (major version×100 plus "sub" version number)
LCOPY	number of "hard" copies for each graphics output - used for movies
LDIAGO	history interval (#)
LDIAGP	display (graphics) interval (#)
LEPW(NTHMX)	local emissivity
LFBO	compartment of origin (1 to nr-1)
LFBT	type of fire (1, 2, ...)
LFMAX	number of intervals in a fire specification
LFPOS	position of the fire in a room (1 to 4) - affects entrainment only
LIMO2	limiting oxygen index in percent (default is 10%)
LPRINT	print interval
MASS(2,NR,NS)	mass in a layer of species ns (1 to ns)
MAXCT	number of entries in the thermal database (max is 57 now)
MFIRET(NS)	mass release rate of species ns - transient
MINMAS	minimum mass in mass(...)
MPRODR(NV,NS)	species production rate for specified fire - see tech ref for details

MPSDAT(3)	date of this run (see also crdate and rundat)
N	number of compartments in use (including the outside)
N2,N3,N4	n+1,2n+1,3n+1
NA(IB)	starting node for branch ib
NBR	number of branches
NCNODE(I)	number of branches coonnected to hvac node i
NCONFG	0 or 1 if a graphics descriptor is present
NDIV(MXSLB,NWAL,NR)	number of interior nodes in a wall material (of mxslb slabs)
NDT	number of ducts
NDUMPR	0 or 1 if a history file specification is present
NE(IB)	exit node number of branch ib
NEUTRAL(NR,NR)	number of neutral planes for a vent - not very useful
NEXT	number of exterior nodes
NF(IB)	0 if duct, fan number if a fan
NFAN	number of fans
NFC(K)	number of polynomial coefficients for fan k
NFT	number of simple fittings
NLIST(NTHMX)	list of thermal names used by the current thermal database
NLSPCT	number of species in this run
NM1	actual number of compartments (N-1)
NNODE	number of nodes in the hvac system
NOPNMX	not used
NRESTR	restart time (0 means no restart)
NRFLOW	not used
NSLB(NWAL,NR)	number of slabs in a particular wall
NSMAX	maximum simulation time (seconds)
NW(NR,NR)	switch for horizontal vents - coded for 1 to 4 by powers of 2
NWV(NR,NR)	switch for vertical vents
ONTARGET(NR)	absolute radiation from the upper layer to a target (less ambient)
P(NT)	solution vector of pressure, upper and lower temperature, volume
PA	ambient pressure at the measured station
PAMB(NR)	ambient pressure in a compartment prior to the fire
PMAX(NT),PMIN(NT)	limits on the values in P(NT)
POFSET	a pressure offset to help solve the stiffness problem
PPMDV(2,NR,NS)	mass concentration (kg/m^3)
PREF	default reference pressure (1.03 e+5)
QC(2,NR)	net convective heat loss from a zone (Watts)
QF(2,NR)	net heat generation rate of a fire into a zone (Watts)
QFIRED(NV)	heat release rate for specified fire
QMAX(K),QMIN(K)	flow rate at hmax(k) and hmin(k)
QR(2,NR)	net radiative loss from a zone (Watts)
QRADRL	fraction of heat which leaves a fire as radiation
RA	default station ambient (inside) density (kg/m^3)
RAMB(NR)	initial (ambient) mass density in a compartment
RELHUM	initial relative humidity (default ->0%)

RGAS	"universal" gas constant
ROHB(IB)	density of gases in branch ib
RR(ID)	relative roughness of walls of duct id
RW(MXSLB,NWAL,NR)	material density of a boundary slab (kg/m^3)
SA(NR,NR,4)	flow field upper to lower (kg/s)
SAL	station elevation (m) - default to zero
SAU(NR,NR,4)	entrainment rate into the upper layer
SIGM	Stefan-Boltzmann constant (5.67×10^{-8} $W/m^2/K^4$)
SS(NR,NR,4)	flow field from upper to upper layer (kg/s)
STIME	current simulation time (s) - corresponds to itmstp
SWITCH(NWAL,NR)	logical switch for wall conduction - switch...,nr) is used for output
TA	station ambient temperature (K)
TAMB(NR)	ambient temperature in a compartment (K)
TBR(IB)	absolute temperature of gases in branch ib
TE	pyrolysis temperature of the fuel
TERRORS(NTHMX)	code for errors in the thermal database
TFIRED(NV)	time interval specification
TFIRET	current time for interpolation
TFMAXT	maximum time for the specified fire
TGIGNT	ignition temperature for a well stirred gas - limits fires in vents
THDEF(NTHMX)	logical for whether thermal name is correctly defined
THRMFILE*20	name of the thermal database
TOXICT(2,NR,NS)	conglomeration of stuff for output - see Tech Ref.
TREF	default reference temperature
TWE(NWAL,NR)	temperature of the gas external to a compartment boundary
TWJ(NWAL,NR,NN)	temperature profile in the boundaries (ceiling, floor, upper/lower wall)
VOL(IBR)	volume of branch ibr
VR(NR)	volume of a compartment
VVAREA(NR,NR)	area of a vertical vent
WINDC(NR)	wind coefficient for a vent facing the outside
WINDPW	wind power law coefficient
WINDRF	wind reference height (m)
WINDV	wind reference velocity at windrf

There is only one variable in the common block moco1b and it is the title of the data file:

TITLE(50) a character*1 The value read from the first line of the data file

B.3 Intermodule communication through PARAMS.INC

The first set are in the unlabelled common, and are the intial and boundary conditions for the physical routines.

EXxx exterior equivalents of ta, pa and ra: exta, expa, exra
ERA(NR),EPA(NR),ETA(NR) exterior equivalents of ramb, pamb, and tamb: era,epa,eta
EXSAL station elevation of the outside of the structure
ALLOWED(NS) which species can be set relative to the fuel
EXSET logical variable set if the external ambient has been set separately from the internal ambient
MAPLTW(NWAL) mapping for the convective and radiative phenomena
QFR(2,NR) total radiative heat gain (+) or loss (-) by a layer
QFC(2,NR) " convective " " " "
QSCNV(NWAL,NR) convective flux to a wall (w/m^2)
QDOUT(NWAL,NR) net heat loss on the back side of a wall (w/m^2)
QSRADW(NWAL,NR) radiative flux to a wall (w/m^2)
QDIN(NWAL,NR) net heat *gain* to an interior side (exposed) of a wall (w/m^2)
QCVENT(NR,NR,4,NV) opening parameter for a vent - this is a fraction which is 0.0->1.0
O2N2(NS) ratio of oxygen to nitrogen in the ambient
HWJDOT(NWAL,NR) rate of deposition of hydrogen chloride to a wall surface ($kg/s/m^2$)

The common block "params" contains interpolation variables. These are set in PYROLS.

HCRATT
ITERPT, ITIME1, ITIME2 counters
TIMEI1, TIMEI2, TIMEI3 actual interpolation values

B.4 Thermophysical properties passed through THERMP.INC

LRW(MXSLB,NTHMX) local mass density of a material slab
LNSLB(NTHMX) local number of slabs in a material (used in reading from the database)
LCW(MXSLB,NTHMX) local heat capacity
LFKW(MXSLB,NTHMX) local conductivity
LFLW(MXSLB,NTHMX) local thickness (m)
CNAME(NWAL,NR) name (pointer) of thermal property specfied for a boundary

B.5 The physical environment set by the data copy routine

The subroutines, SOLVE and DIFEQ, use the array, P, to store the physical quantities that are being modeled. This array contains the values of pressure, upper layer volume, upper/lower layer temperature, upper/lower layer gas species mass and wall species density for each compartment in the simulation. This structure is illustrated in figure 10. The solver array, Y, in CCFM.VENTS has a similar structure.

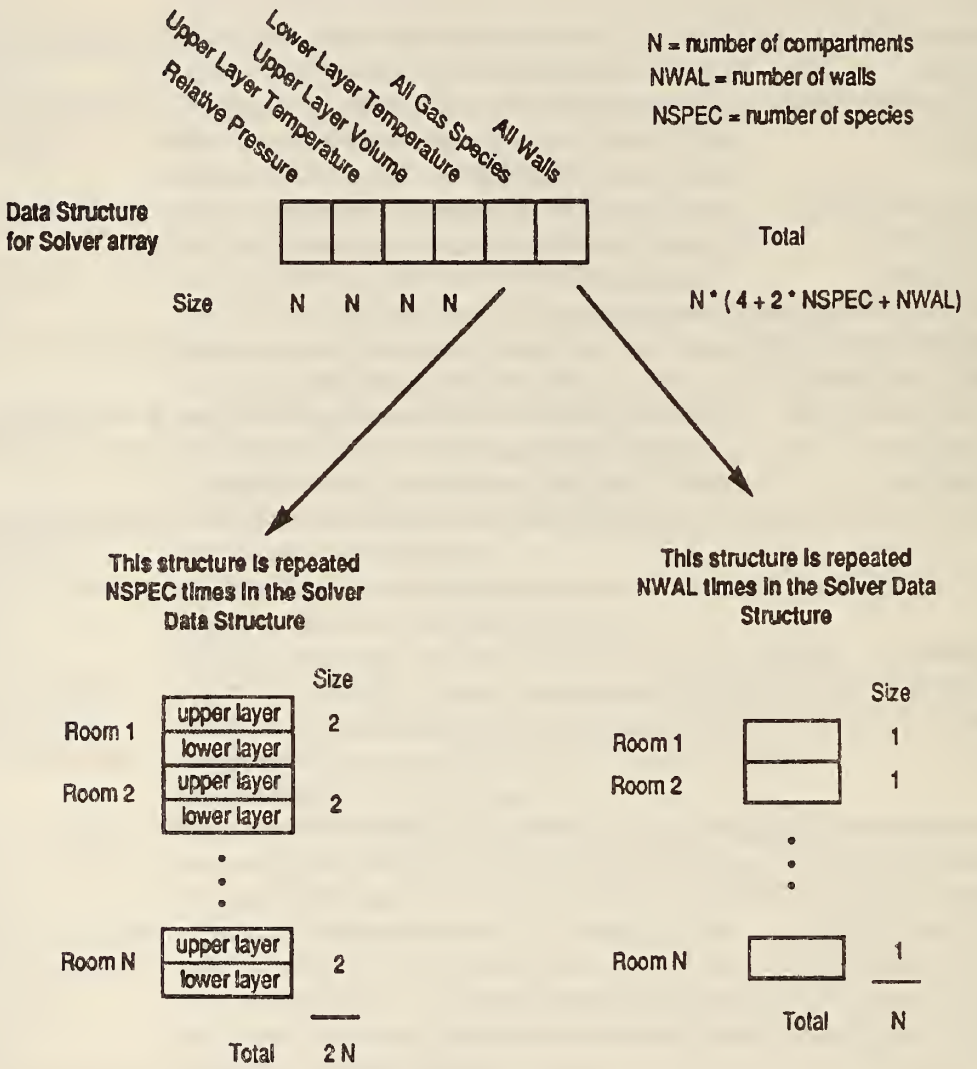

Figure 10. Solver Array Structure.

Many subroutines[6] in CFAST require the information contained in the P array. If this information is obtained directly, i.e. by accessing the P array in an assignment statement, then whenever the structure of P changes, modifications will have to be made throughout CFAST. Some model changes that would alter the structure of the P array are use of layer mass instead of layer temperature, use of the layer interface height instead of upper layer

6 The subroutines that use P may be identified by using the tool **comcheck**. This is how the DATACOPY routine was implemented in CFAST.

volume or a change in the order of the variables. In order to minimize the work required to implement such a modification in the structure of P, a data copy routine was written. This routine copies data from the P array to a set of environment variables that physical routines may then access. These variables are distinguished from other CFAST variables by beginning with 'ZZ'. So whenever a change is made in P, only DATACOPY need be re-written. DATACOPY is called by DSOURC and SOLVE to update the environment variables for a given value of P.

Certain environment variables do not change througout a simulation. They do not need to be updated each time DATACOPY is called. These variables end in 'MAX' or 'MIN.' DATACOPY is called one time by NPUTP to set these variables.

DATACOPY is called with two arguments. The first argument is the solver array, P. The second argument is a flag that determines which environment variables are set. If CONSTVAR is passed then time independent variables are set. If ODEVARA is passed then environment variables associated with pressure, layer volume or layer temperatures are set. If ODEVARB is passed then the environment variables associated with gas or wall species are set.

CONSTVAR, ODEVARA and ODEVARB are integer parameter variables defined in CENVIRO.INC and set to 1, 2, 4 respectively. The subroutine call

```
CALL DATACOPY(P,ODEVAR+ODEVARB)
```

causes all time dependent environment variables to be set. The subroutine call

```
CALL DATACOPY(P,CONSTVAR+ODEVAR+ODEVARB)
```

causes all environment variables to be set. If CONSTVAR is the only flag passed to DATACOPY in a call as in

```
CALL DATACOPY(DUMMY,CONSTVAR)
```

then the first argument is not accessed within DATACOPY. So it can be any "dummy" variable, *i.e.*, it does not have to be an array.

Time Independent Environment Variables - Set with CALL DATACOPY(P,CONSTVAR)

ZZTMIN(I)	minimum temperature in I'th room
ZZTMAX(I)	maximum temperature in I'th room
ZZVMIN(I)	minimum volume in I'th room
ZZVMAX(I)	maximum volume in I'th room
ZZPMIN(I)	minimum pressure in I'th room
ZZPMAX(I)	maximum pressure in I'th room

ZZGMAX(I)	maximum gas species amount in I'th room
ZZGMIN(I)	minimum gas species amount in I'th room
ZZWMAX(I)	maximum wall species amount in I'th room
ZZWMIN(I)	minimum wall species amount in I'th room

Time Dependent Environement Variables - Set with CALL DATACOPY(P,ODEVARA)

The second index in the following environment variables denotes the layer. The upper layer has an index of 1 while the lower layer has an index of 2. Parameters, UPPER and LOWER are defined to 1 and 2 respectively in the include file CPARAMS.INC to make environment variable access easier. For example, ZZTEMP(3,LOWER) contains the temperature of the lower layer in the third room.

ZZVOL(I,L)	volume of L'th layer in I'th room
ZZHLAY(I,L)	height of L'th layer in I'th room
ZZRELP(I)	pressure relative to POFSET in I'th room
ZZPABS(I)	absolute pressure in I'th room
ZZTEMP(I,L)	temperature of L'th layer in I'th room
ZZRHO(I,L)	density of L'th layer in I'th room
ZZMASS(I,L)	mass of L'th layer in I'th room
ZZQ(I,L)	energy of L'th layer in I'th room
ZZTIME(I)	CPU time in seconds for I'th process.

Time Dependent Environment Variables - Set with CALL DATACOPY(P,ODEVARB)

ZZGSPEC(I,L,S)	concentration of S'th species in L'th layer in I'th room
ZZWSPEC(I,W)	concentration of species (currently HCl) in W'th wall in I'th room

B.6 Shell and program environment, CSHELL.INC

RUNDAT(3)	today's date (day, month, year) rundat is copied to mpsdatas soon as the model kernel is started. done in initfs and cfast body.
ADVFEA	logical switch for availability of advanced features in the data editor, such as ability to run the model from the data editor, set colors, ...
SHELL	logical - set to true is the shell is doing the procedure
CF_EXIST	logical - set to true if the named configuration file exists
HEADER	name used in the first six characters of the configuration file
VERSION	actual working version of the model - copied to ivers by initfs or cfast
DLEN	number of characters in the database path - zero implies not used
PLEN	number of characters in the data path - zero implies not used

UNITS(7)	switch for units in sunit and cunit - see the header in those routines for meanings of the values
ICHDR,ICSUB,ICTXT,ICPRO,ICMSG,ICHLP,ICBG,ICMBG,ICHBG,ICEBG,ICEMS	color settings used for the set and edit modules
IOFILI, IOFILO	input and output unit numbers - defined in the "initcs" block data
THRMFILE*60	name of the thermophysical properties file
CONFIG*60	name of the configuration file
OUTFILE*60	name of the output file (if appropriate)
NNFILE*60	name of the input file (if appropriate)
DFILE*60	data file read in from the configuration file
RFILE*60	restart file
OPTIONS*4	MOPT allowed options (see the discussion in section 4)
CARDTYPE*4	type of graphics adaptor
OFILE*60	name of the file containing other objects which can burn
CURRENT*64	current path, which can be different than the database and data paths
PATH*64	path where the data files are found
DPATH*64	path for executables and databases
MPSDATC*8	today's date in character format (see rundat)
GFILE,PFILE*60	geometry file - not used in this version - will be for comparment interconnections
DUMPF*60	history file - set by nputq
PASSFILE*60	used by the shell routine to pass the environment to cfast, cf_plot, ...

B.7 Setting the precision for the model, PRECIS.INC

This is not a common block. Rather it is the header used to set the precision of all real (as opposed to integer) variables which are not strongly typed. See chapter 3, and in particular section 3.3 for the use that is made of this header. It should preceed and specification lines in any routine that will be performing physical calculations. All graphics routines used single precision, and these variables should be so typed.

```
%IF DOUBLE
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
%ELSE
    IMPLICIT REAL (A-H,O-Z)
%ENDIF
```

Appendix C: DEPENDENCY CHART OF ROUTINES - WHO CALLS WHOM

This appendix contains information on how each subroutine and common block in CFAST are related to each other. This information is essential to the person who wishes to modify CFAST. For example, if one wishes to change the number of arguments in the calling list of a subroutine one would need to know what subroutines call it.

The main heading, NAME, is given in **bold text**. Each entry has a description and up to four sub-headings: CALLS, LIB,COMMONS and CALLED BY. The subheadings CALLS and LIB are similar. They both list external references to NAME., *i.e.*, what routines NAME calls. The source code for the routines that are listed under CALLS appear in the same computer file as NAME. On the other hand, the routines listed by the LIB sub-heading do not. Some examples of routines that would appear under LIB are FORTRAN supplied functions such as ABS, SQRT, MOD, etc. The names listed next to COMMONS are the COMMON blocks that appear in the routine NAME. The routines that are listed next to CALLED BY are those routine that call NAME.

NAME ... description of NAME ...

CALLS: SUB1, SUB2, ...
LIB: SUBA, SUBB, ...
COMMONS: COM1, COM2, ...
CALLED BY: SUBa, SUBb, ...

```
************************************
***  PROGRAM MAP (1 LEVEL DEEP)  ***
************************************

*** SOURCE ROUTINES ***

ROUTINE: ATMOSP
         CALLED BY: INITAMB

ROUTINE: BSTRNG
         CALLED BY: CMDLINE

ROUTINE: BXBLIT
         LIB:       COLOR    FILTYP   PLYGON
         CALLED BY: DISPLAY

ROUTINE: CFAST
         CALLS:     CPTIME   DISCLAIM DISPLAY  DUMPER   INITMM   INITSPEC NPUTO
                    NPUTP    NPUTT    OPENSHEL READOP   RESTRT   RESULT   SOLVE
         LIB:       DIFST    ENDDIS   FLOAT    IFIX
         COMMONS:   BLANK CO CENVIRO  CONFG1   CONFG2    INPUT1   INPUT2   MOCO1A
                    MOCO1B   MOCO1D   MOCO1E   MSKBD    PARAMS
         CALLED BY: NONE - NO ROUTINES CALL CFAST
```

```
ROUTINE: CHEMIE
         LIB:       EXP       MAX       MIN
         COMMONS:   MOCO1A    MOCO1B
         CALLED BY: DSOURC

ROUTINE: CMDLINE
         CALLS:     BSTRNG
         LIB:       GETCL
         CALLED BY: READOP

ROUTINE: CNDUCT
         CALLS:     CONVEC
         LIB:       ABS       MAX
         COMMONS:   MOCO1A    MOCO1B    MOCO1D    MOCO1E
         CALLED BY: SOLVE

ROUTINE: CONSCHEK
         COMMONS:   CENVIRO   MOCO1A    MOCO1B
         CALLED BY: NONE - NO ROUTINES CALL CONSCHEK

ROUTINE: CONVEC
         LIB:       ABS
         COMMONS:   MOCO1A    MOCO1B
         CALLED BY: CNDUCT    DSOURC

ROUTINE: CONVRT
         CALLS:     DATYPE
         LIB:       MIN
         CALLED BY: READCV    READIN

ROUTINE: CPTIME
         LIB:       TIMER
         CALLED BY: CFAST     SOLVE

ROUTINE: DATACOPY
         LIB:       MAX       MOD
         COMMONS:   CENVIRO   MOCO1A    MOCO1B
         CALLED BY: DSOURC    INITAMB   INITSPEC SOLVE

ROUTINE: DATYPE
         CALLED BY: CONVRT

ROUTINE: DIFEQ
         CALLS:     DIFERR
         LIB:       ABS       DFE       MAX       MIN       SQRT
         CALLED BY: SOLVE

ROUTINE: DIFERR
         CALLED BY: DIFEQ

ROUTINE: DISCLAIM
         COMMONS:   CONFG1    CONFG2
         CALLED BY: CFAST

ROUTINE: DISPLAY
         CALLS:     BXBLIT    GETVIEW   GRAFIT    INIPAR    OUTPU1    PALETTE   RESETLUT
                    TOXICB    TOXICH    TOXICR    VWPRT
         LIB:       BOXPLT    COLOR     DEF3D     DEVICE    ENDFRM    FLOAT     FRAME
                    GRILAB    HDCOPY    IFIX      INT       LABEL     LINWID    MAX
                    MIN       MOD       MOUSE_OF  NEWFRM    PLOTLN    VIEWTR
         COMMONS:   CENVIRO   DFLTS     MOCO1A    MOCO1B    MOCO2A    MOCO2B
         CALLED BY: CFAST     SOLVE

ROUTINE: DISRAD
```

```
          LIB:        MAX       MIN
          COMMONS:    MOCO1A    MOCO1B
          CALLED BY:  DSOURC

ROUTINE:  DISTHE
          COMMONS:    CONFG1    CONFG2    MOCO1A    MOCO1B    MOCO1D    MOCO1E
          CALLED BY:  NPUTO

ROUTINE:  DREADIN
          CALLED BY:  RESTRT

ROUTINE:  DSOURC
          CALLS:      CHEMIE    CONVEC    DATACOPY  DISRAD    ENTRFL    FIRPLM    FIRRAD
                      FLOW      HCLTRAN   PYROLS    STPORT
          LIB:        IAND      MAX       MIN
          COMMONS:    BLANK CO  CENVIRO   MOCO1A    MOCO1B    PARAMS
          CALLED BY:  NONE - NO ROUTINES CALL DSOURC

ROUTINE:  DUMPER
          CALLS:      LENOCO    WRITEOT
          LIB:        CHAR      MOD
          COMMONS:    CONFG1    CONFG2    INPUT1    INPUT2    MOCO1A    MOCO1B    MSKBD
          CALLED BY:  CFAST     SOLVE

ROUTINE:  ENTRFL
          LIB:        MAX
          COMMONS:    MOCO1A    MOCO1B
          CALLED BY:  DSOURC

ROUTINE:  FIRPLM
          LIB:        MAX
          CALLED BY:  DSOURC

ROUTINE:  FIRRAD
          LIB:        MAX
          COMMONS:    MOCO1A    MOCO1B
          CALLED BY:  DSOURC

ROUTINE:  FLOW
          CALLS:      FRFLOW
          LIB:        MAX       MIN
          COMMONS:    BLANK CO  CENVIRO   MOCO1A    MOCO1B    PARAMS
          CALLED BY:  DSOURC

ROUTINE:  FLWOUT
          COMMONS:    MOCO1A    MOCO1B
          CALLED BY:  RESULT

ROUTINE:  FRFLOW
          CALLS:      TTCB
          LIB:        MAX       MIN       SQRT
          CALLED BY:  FLOW

ROUTINE:  GASLOAD
          CALLS:      READCV
          COMMONS:    MOCO2A    MOCO2B    READ1C    READ2C
          CALLED BY:  LOADIN

ROUTINE:  GETVIEW
          LIB:        PLYGNS    PLYPLT
          COMMONS:    MOCO2A    MOCO2B
          CALLED BY:  DISPLAY

ROUTINE:  GRAFIT
          LIB:        ABS       BOXPLT    CHPLOT    DEFINE    FLOAT     FNUMBR    INT
```

```
                LABEL    LINE     LINWID   LNPLOT   MIN      PLYGON   SIGN
                WDCOUNT
        COMMONS:  DEVTYP   GRFTYP
        CALLED BY: DISPLAY

ROUTINE: GRQUERY
        LIB:      IAND     QUERYGRA
        COMMONS:  CONFG1   CONFG2
        CALLED BY: NONE - NO ROUTINES CALL GRQUERY

ROUTINE: HCLTRAN
        LIB:      ABS      EXP      LOG      MAX
        COMMONS:  BLANK CO CENVIRO  MOCO1A   MOCO1B   PARAMS
        CALLED BY: DSOURC

ROUTINE: HSETS
        LIB:      IAND     ISHFT    MOD
        COMMONS:  GRFTYP
        CALLED BY: WDDRAW

ROUTINE: HVAC
        CALLS:    HVFREX   HVMFLO   HVSFLO   HVTOEX
        COMMONS:  MOCO1A   MOCO1B
        CALLED BY: SOLVE

ROUTINE: HVFAN
        LIB:      FLOAT
        COMMONS:  MOCO1A   MOCO1B
        CALLED BY: HVMFLO

ROUTINE: HVFREX
        LIB:      MIN
        COMMONS:  CENVIRO  MOCO1A   MOCO1B
        CALLED BY: HVAC

ROUTINE: HVFRIC
        LIB:      ABS
        CALLED BY: HVMFLO

ROUTINE: HVINIT
        LIB:      FLOAT    MAX      MIN
        COMMONS:  BLANK CO CENVIRO  MOCO1A   MOCO1B   PARAMS
        CALLED BY: NPUTP

ROUTINE: HVMFLO
        CALLS:    HVFAN    HVFRIC   HVVIS
        LIB:      ABS      MAX      SIGN     SQRT
        COMMONS:  MOCO1A   MOCO1B
        CALLED BY: HVAC

ROUTINE: HVSFLO
        LIB:      MIN
        COMMONS:  MOCO1A   MOCO1B
        CALLED BY: HVAC

ROUTINE: HVTOEX
        COMMONS:  CENVIRO  MOCO1A   MOCO1B
        CALLED BY: HVAC

ROUTINE: HVVIS
        LIB:      FLOAT
        CALLED BY: HVMFLO

ROUTINE: INIPAR
```

```
        LIB:       F_RCOUNT
        COMMONS:   DFLTS    PLTFLT   PLTPAR
        CALLED BY: DISPLAY

ROUTINE: INITAMB
        CALLS:     ATMOSP   DATACOPY
        LIB:       MAX      MIN
        COMMONS:   BLANK CO CENVIRO  MOCO1A   MOCO1B   PARAMS
        CALLED BY: NPUTP

ROUTINE: INITBK
        COMMONS:   INPUT1   INPUT2   MSKBD
        CALLED BY: NONE - NO ROUTINES CALL INITBK

ROUTINE: INITCS
        COMMONS:   CONFG1   CONFG2
        CALLED BY: NONE - NO ROUTINES CALL INITCS

ROUTINE: INITDI
        COMMONS:   DFLTS    MOCO2A   MOCO2B
        CALLED BY: NONE - NO ROUTINES CALL INITDI

ROUTINE: INITMM
        COMMONS:   BLANK CO CONFG1   CONFG2   INPUT1   INPUT2   MOCO1A   MOCO1B
                   MOCO1D   MOCO1E   MSKBD    PARAMS
        CALLED BY: CFAST

ROUTINE: INITSPEC
        CALLS:     DATACOPY TOXIC
        COMMONS:   BLANK CO CENVIRO  CONFG1   CONFG2   MOCO1A   MOCO1B   MOCO1D
                   MOCO1E   PARAMS
        CALLED BY: CFAST

ROUTINE: LENOCO
        CALLED BY: DUMPER   RESTRT

ROUTINE: LOADIN
        CALLS:     GASLOAD  LOADUP   READAS   READCV
        COMMONS:   DFLTS    MOCO1A   MOCO1B   MOCO2A   MOCO2B   READ1C   READ2C
        CALLED BY: NPUTP

ROUTINE: LOADUP
        CALLS:     READCV
        COMMONS:   READ1C   READ2C
        CALLED BY: LOADIN

ROUTINE: MVOUT
        CALLED BY: NPUTO

ROUTINE: NPUTO
        CALLS:     DISTHE   MVOUT
        LIB:       IAND     ISHFT    MVOLAST
        COMMONS:   BLANK CO CONFG1   CONFG2   MOCO1A   MOCO1B   PARAMS
        CALLED BY: CFAST

ROUTINE: NPUTP
        CALLS:     HVINIT   INITAMB  LOADIN   NPUTQ
        LIB:       ABS      IABS     MOD
        COMMONS:   BLANK CO CONFG1   CONFG2   MOCO1A   MOCO1B   PARAMS
        CALLED BY: CFAST

ROUTINE: NPUTQ
        CALLS:     READIN
        LIB:       IAND     IOR      MAX      MIN      READBF   READFL   READRS
        COMMONS:   BLANK CO CONFG1   CONFG2   INPUT1   INPUT2   MOCO1A   MOCO1B
```

```
                MOCO1D   MOCO1E   MSKBD    PARAMS
        CALLED BY: NPUTP

ROUTINE: NPUTT
        CALLS:     READAS   READIN   SSTRNG
        LIB:       CHAR     FLOAT    INT      MAX      MIN      SQRT
        COMMONS:   CONFG1   CONFG2   INPUT1   INPUT2   MOCO1A   MOCO1B   MOCO1D
                   MOCO1E   MSKBD    READ1C   READ2C
        CALLED BY: CFAST

ROUTINE: OPENSHEL
        CALLS:     READCF   SSTRNG
        COMMONS:   CONFG1   CONFG2
        CALLED BY: CFAST

ROUTINE: OUTPU1
        COMMONS:   CENVIRO  MOCO1A   MOCO1B   PLTPAR
        CALLED BY: DISPLAY

ROUTINE: PALETTE
        LIB:       COLOR    FILTYP   LABEL    PLYGON
        CALLED BY: DISPLAY

ROUTINE: PYROLS
        LIB:       MIN
        COMMONS:   MOCO1A   MOCO1B
        CALLED BY: DSOURC

ROUTINE: READAS
        COMMONS:   CONFG1   CONFG2
        CALLED BY: LOADIN   NPUTT    READIN

ROUTINE: READCF
        COMMONS:   CONFG1   CONFG2
        CALLED BY: OPENSHEL

ROUTINE: READCV
        CALLS:     CONVRT   SSTRNG
        LIB:       FLOAT    IFIX
        COMMONS:   READ1C   READ2C
        CALLED BY: GASLOAD  LOADIN   LOADUP

ROUTINE: READIN
        CALLS:     CONVRT   READAS   SSTRNG
        LIB:       FLOAT    IFIX     MIN
        COMMONS:   READ1C   READ2C
        CALLED BY: NPUTQ    NPUTT

ROUTINE: READOP
        CALLS:     CMDLINE  TOUPPER
        LIB:       DATE
        COMMONS:   CONFG1   CONFG2
        CALLED BY: CFAST

ROUTINE: RESETLUT
        CALLED BY: DISPLAY

ROUTINE: RESTRT
        CALLS:     DREADIN  LENOCO
        LIB:       MOD
        COMMONS:   MOCO1A   MOCO1B
        CALLED BY: CFAST

ROUTINE: RESULT
```

```
          CALLS:      FLWOUT
          LIB:        IAND      ISHFT     MAX
          COMMONS:    CENVIRO   MOCO1A    MOCO1B
          CALLED BY:  CFAST     SOLVE

ROUTINE:  SETLUT
          LIB:        LOADPALE  SETPALET
          CALLED BY:  NONE - NO ROUTINES CALL SETLUT

ROUTINE:  SOLVE
          CALLS:      CNDUCT    CPTIME    DATACOPY  DIFEQ     DISPLAY   DUMPER    HVAC
                      RESULT    TOXIC
          LIB:        FLOAT     GRABKY    MOD
          COMMONS:    BLANK CO  CENVIRO   MOCO1A    MOCO1B    PARAMS
          CALLED BY:  CFAST

ROUTINE:  SSTRNG
          CALLED BY:  NPUTT     OPENSHEL  READCV    READIN

ROUTINE:  STPORT
          LIB:        IAND      MAX
          COMMONS:    BLANK CO  CENVIRO   MOCO1A    MOCO1B    PARAMS
          CALLED BY:  DSOURC

ROUTINE:  TOUPPER
          LIB:        CHAR      ICHAR
          CALLED BY:  READOP

ROUTINE:  TOXIC
          LIB:        MAX
          COMMONS:    BLANK CO  CENVIRO   MOCO1A    MOCO1B    PARAMS
          CALLED BY:  INITSPEC  SOLVE

ROUTINE:  TOXICB
          LIB:        COLOR     FLOAT     LABEL
          COMMONS:    MOCO1A    MOCO1B    MOCO2A    MOCO2B
          CALLED BY:  DISPLAY

ROUTINE:  TOXICH
          LIB:        CHRSET    COLOR     FLOAT     LABEL     LINE
          COMMONS:    MOCO1A    MOCO1B
          CALLED BY:  DISPLAY

ROUTINE:  TOXICR
          LIB:        COLOR     FILTYP    FLOAT     LABEL     PLYGON
          COMMONS:    MOCO1A    MOCO1B    MOCO2A    MOCO2B
          CALLED BY:  DISPLAY

ROUTINE:  TTCB
          LIB:        EXP       MIN
          CALLED BY:  FRFLOW

ROUTINE:  VWPRT
          COMMONS:    MOCO1A    MOCO1B    MOCO2A    MOCO2B
          CALLED BY:  DISPLAY

ROUTINE:  WDDRAW
          CALLS:      HSETS
          LIB:        ABS       COLOR     COS       HHDRAW    ICHAR     IDCHAR    SIN
          COMMONS:    DEVTYP    GRFTYP
          CALLED BY:  NONE - NO ROUTINES CALL WDDRAW

ROUTINE:  WRITEOT
          CALLED BY:  DUMPER
```

*** LIBRARY ROUTINES ***

ROUTINE: **ABS**
CALLED BY: CNDUCT CONVEC DIFEQ GRAFIT HCLTRAN HVFRIC HVMFLO
NPUTP WDDRAW

ROUTINE: **BOXPLT**
CALLED BY: DISPLAY GRAFIT

ROUTINE: **CHAR**
CALLED BY: DUMPER NPUTT TOUPPER

ROUTINE: **CHPLOT**
CALLED BY: GRAFIT

ROUTINE: **CHRSET**
CALLED BY: TOXICH

ROUTINE: **COLOR**
CALLED BY: BXBLIT DISPLAY PALETTE TOXICB TOXICH TOXICR WDDRAW

ROUTINE: **COS**
CALLED BY: WDDRAW

ROUTINE: **DATE**
CALLED BY: READOP

ROUTINE: **DEF3D**
CALLED BY: DISPLAY

ROUTINE: **DEFINE**
CALLED BY: GRAFIT

ROUTINE: **DEVICE**
CALLED BY: DISPLAY

ROUTINE: **DFE**
CALLED BY: DIFEQ

ROUTINE: **DIFST**
CALLED BY: CFAST

ROUTINE: **ENDDIS**
CALLED BY: CFAST

ROUTINE: **ENDFRM**
CALLED BY: DISPLAY

ROUTINE: **EXP**
CALLED BY: CHEMIE HCLTRAN TTCB

ROUTINE: **FILTYP**
CALLED BY: BXBLIT PALETTE TOXICR

ROUTINE: **FLOAT**
CALLED BY: CFAST DISPLAY GRAFIT HVFAN HVINIT HVVIS NPUTT
READCV READIN SOLVE TOXICB TOXICH TOXICR

ROUTINE: **FNUMBR**
CALLED BY: GRAFIT

ROUTINE: **FRAME**
CALLED BY: DISPLAY

```
ROUTINE: F_RCOUNT
         CALLED BY: INIPAR

ROUTINE: GETCL
         CALLED BY: CMDLINE

ROUTINE: GRABKY
         CALLED BY: SOLVE

ROUTINE: GRILAB
         CALLED BY: DISPLAY

ROUTINE: HDCOPY
         CALLED BY: DISPLAY

ROUTINE: HHDRAW
         CALLED BY: WDDRAW

ROUTINE: IABS
         CALLED BY: NPUTP

ROUTINE: IAND
         CALLED BY: DSOURC   GRQUERY  HSETS    NPUTO    NPUTQ    RESULT   STPORT

ROUTINE: ICHAR
         CALLED BY: TOUPPER  WDDRAW

ROUTINE: IDCHAR
         CALLED BY: WDDRAW

ROUTINE: IFIX
         CALLED BY: CFAST    DISPLAY  READCV   READIN

ROUTINE: INT
         CALLED BY: DISPLAY  GRAFIT   NPUTT

ROUTINE: IOR
         CALLED BY: NPUTQ

ROUTINE: ISHFT
         CALLED BY: HSETS    NPUTO    RESULT

ROUTINE: LABEL
         CALLED BY: DISPLAY  GRAFIT   PALETTE  TOXICB   TOXICH   TOXICR

ROUTINE: LINE
         CALLED BY: GRAFIT   TOXICH

ROUTINE: LINWID
         CALLED BY: DISPLAY  GRAFIT

ROUTINE: LNPLOT
         CALLED BY: GRAFIT

ROUTINE: LOADPALE
         CALLED BY: SETLUT

ROUTINE: LOG
         CALLED BY: HCLTRAN

ROUTINE: MAX
         CALLED BY: CHEMIE   CNDUCT   DATACOPY DIFEQ    DISPLAY  DISRAD   DSOURC
                    ENTRFL   FIRPLM   FIRRAD   FLOW     FRFLOW   HCLTRAN  HVINIT
                    HVMFLO   INITAMB  NPUTQ    NPUTT    RESULT   STPORT   TOXIC
```

```
ROUTINE: MIN
         CALLED BY: CHEMIE   CONVRT   DIFEQ    DISPLAY  DISRAD   DSOURC   FLOW
                    FRFLOW   GRAFIT   HVFREX   HVINIT   HVSFLO   INITAMB  NPUTQ
                    NPUTT    PYROLS   READIN   TTCB

ROUTINE: MOD
         CALLED BY: DATACOPY DISPLAY  DUMPER   HSETS    NPUTP    RESTRT   SOLVE

ROUTINE: MOUSE_OF
         CALLED BY: DISPLAY

ROUTINE: MVOLAST
         CALLED BY: NPUTO

ROUTINE: NEWFRM
         CALLED BY: DISPLAY

ROUTINE: PLOTLN
         CALLED BY: DISPLAY

ROUTINE: PLYGNS
         CALLED BY: GETVIEW

ROUTINE: PLYGON
         CALLED BY: BXBLIT   GRAFIT   PALETTE  TOXICR

ROUTINE: PLYPLT
         CALLED BY: GETVIEW

ROUTINE: QUERYGRA
         CALLED BY: GRQUERY

ROUTINE: READBF
         CALLED BY: NPUTQ

ROUTINE: READFL
         CALLED BY: NPUTQ

ROUTINE: READRS
         CALLED BY: NPUTQ

ROUTINE: SETPALET
         CALLED BY: SETLUT

ROUTINE: SIGN
         CALLED BY: GRAFIT   HVMFLO

ROUTINE: SIN
         CALLED BY: WDDRAW

ROUTINE: SQRT
         CALLED BY: DIFEQ    FRFLOW   HVMFLO   NPUTT

ROUTINE: TIMER
         CALLED BY: CPTIME

ROUTINE: VIEWTR
         CALLED BY: DISPLAY

ROUTINE: WDCOUNT
         CALLED BY: GRAFIT
```

Appendix D: INTERFACE PROTOCOL - ROUTINES ALPHABETICALLY LISTING FUNCTION

The following is an annotated list of routines used in the CFAST suite. If a routine is specific to one topic, then it is listed at the end of the line (see the section on the directory structure). Otherwise, it is a general purpose routine. If the module is a main program, then the term "main" is used

ADD_COMOMP	add a comparment to the list (editor)
ADD_THRM	add a material to the tpp file (editor)
ADD_VENT	add a vent to the geometry (editor)
ATMOSP	calculate the ambient atmosphere
AUTOSC	automatic scaling of the axes
AXSCAL	manual axes scaling (plot)
BSTRNG	string search - see sstrng and reference [3]
BUBBLE	bubble sort (plot)
BXBLIT	bit blitter - used for erasing parts of a screen (model)
C1R	generate a single compartment case (input)
C3R	generate a three compartment case (input)
CFAST	main model (model)
CF_IN	data editor (editor)
CF_PLOT	plotting routine (plot)
CF_RPT	report generator (report)
CF_SET	set colors, units, ... (editor)
CHECKA	check validity of a character string (editor)
CHECKI	check validity of an integer (editor)
CHECKR	check validity of a floating point number (editor)
CHEMIE	explicit species generation
CHGUNT	change the effective units (editor)
CKDUP	check for duplicates in the list of variables (plot)
CLEARS	clear the screen
CMDLINE	read and interpret the command line
CNDUCT	heat conduction through objects
CONSCHEK	do conservation summations - compare over time
CONVEC	convective heat loss or gain
CONVRT	convert an ascii string to an integer and floating point number
CPTIME	return the total calculation (not simulation) time since the beginning
CSHELL	shell for CFAST (main)
CUNITS	part of chgunt - set the units (input)
DATACOPY	copy the solver variables to the environment common blocks
DATYPE	figure out what the type of the ascii string is (integer,...)
DEFAULT	list the defaults for plotting (plot)
DEL_COMP	delete a compartment from the list (editor)
DEL_THRM	delete a thermophysical property from the data base (editor)
DEL_VENT	delete a vent from the geometry section

DIFEQ	solver
DIFERR	error stop for difeq
DISAMB	display the ambient (editor)
DISCAL	display the calculation screen (editor)
DISCLAIM	disclaimer notice (not available)
DISFIR	display the fire specification (editor)
DISGEN	overview (editor)
DISGEO	geometry of the model (editor)
DISHCL	hydrogen chloride output (editor)
DISPLAY	graphics output (model)
DISPRM	display the permanent colors, units, ... (editor)
DISRAD	divide up the radition into upper and lower zones
DISRE1	type one results (editor)
DISRE2	type two results (sub menu of disre1 in the editor)
DISTH1	display thermophysical names used (editor)
DISTH2	display thermophysical database (sub menu od disth1 in the editor)
DISTHE	print combined thermophysical data (disth1 and disth2 for the model)
DORAPID	read channels from aa RAPID type data file (plot)
DOTENAB	read tenability history files (plot)
DO_CVENT	modify the CVENT parameters (editor)
DREADIN	read a record (binary) of the history files - called by restrt
DSOURC	calculate the right hand side of the predictive equations
DUMPER	write to the history file
ENTRFL	calculate vent entrainment
ERRMSG	write an error message on the next to last line of the screen (editor)
FFILE	grab a list of files, and choose one editor [3]
FILESORT	sort the list of files by name and extension - simple bubble sort (editor)
FIRPLM	calculate the size, *etc.* of the plume from a fire
FIRRAD	radiation transfer
FLOW,FRFLOW	vent flow
FLWOUT	display the flow field (model)
F_INFO	get the version, run date, *etc.* of a dump (history) file
F_INFO10	used by f_info - get the extra data for version 1.0 of CFAST
GASLOAD	read species information for the display routines (model)
GETLEG	get a legend lengthfor a plot (plot)
GETVIEW	read a three dimensional view for display (model)
GOPAGE	transfer to a section in the editor
GRAFIT	modified version of GRAFIT from DEVICE [10] (model)
GRAFIT2	modified version of GRAFIT (plot)
GRQUERY	get the graphics adaptor type
GTRAILER	write a simple graphics descriptor file
HCLTRAN	hydrogen chloride deposition to walls
HEADING	display top line of the edit screen (editor)
HELP	get some help
HLPINS	check for available keywords (editor)
HLPTXT	read the help file (unit 2), (editor)
HLPXTND	extended help information (editor)
HSETS	read the stroke definitions for characters from DEVFONT.nnn

HVAC	run the mechanical ventilation model
HVFAN	calculate a flow based on a fan curve
HVFREX	data copy from the compartment format into the ventilation format
HVFRIC	duct friction factor
HVINIT	initialize the mechanical ventilation data structure
HVMFLO	mass flow in the mechanical ventilation system
HVSFLO	species flow in the mechanical ventilation system
HVTOEX	data copy to the compartment format from the ventilation format
HVVIS	viscosity of air
IMOUSE	mouse function call (in mouse.for)
INIPAR	labels, titles, units, and so on for graphics display routines
INITAMB	initialize the ambient conditions based on TAMB and EAMB
INITFS	initialization routine (editor)
INITMM	initialize main memory
INITSPEC	initialize the species, based on the ambient conditions
LENGTH	function to calculate the length of a character string (plot)
LENOCO	length of the main common block (moco1a) in words
LENOFF	display the length of the common block (main)
LISTDEF	list the default settings (plot)
LISTOP	list options (plot)
LIST_CF	display the current configuration file data (main)
LIST_TPP	list the thermophysical properties file (main)
LOADIN	read and sort out the graphics descriptors
LOADUP	read and sort out the graphics descriptors
LOWERC	convert ASCII characters to lower case
LSTFIL	display a file name (actually and ascii string) (editor)
LSTRNG	length of a string - similar to LENGTH (editor)
LSTVAL	display a line (row) of values (editor)
LUNITS	set up the units matrix for conversion by CUNITS
MAKECF	write a configuration file
MESS	write a string to IOFILO
MESSNS	screen write in a window (MESSNR writes at the current cursor location)
MSGNR	send a message to IOFILO without a <cr><lf> pair
MODTIM	modify the time intervals (editor)
MOUSE	grabms and all the other routines for accessing a mouse or trackball
MVOUT	display the mechanical ventilation information
NAILED	write to IOFILO the tenability information for people nailed in place (report)
NEWFILE	get the name of a data file (editor)
NPUTKB	read the key board with messages displayed; filter special keys
NPUTO	write to IOFILO the general output (model)
NPUTOR	same as NPUTO but specialized for the report generator (report)
NPUTP	initialize the data in the model structures
NPUTQ	read the main data file
NPUTT	read the thermophysical properties file
OPENFILE	open a history file for reading, and get the version and creation date
OPENHELP	open the help file (editor)
OPENSHEL	read the configuration file and set up the environment
OUTFIL	display the list of configuration, data,... files (editor)

OUTI	display a row of integer data (editor)
OUTPU	get a list of information from the main data structure
OUTPU1	modified form of OUTPU - a single piece of data (model)
OUTPU10	used by OUTPU (not OUTPU1) to return version 1.0 data
OUTR	display a row of floating point data (editor)
PALETTE	show the palette for the binary decision trees (model)
PLOTIT	simple text based plot (editor)
POLNUM	insert the compartment number in a 3D display of an object (model)
POSIT	cursor position for menus (editor)
PRMCLR	set the permanent colors (editor)
PRNTLIST	display a list of variables that can be accessed (plot)
PRNTVALS	list values of variables in the list of channels (plot)
PUTLEG	put the legend on a plot (plot)
PYROLS	interpolating routine
QUITFI	quit, but check for modified files first (editor)
RANGE	filter the range of data based on the units (editor)
RDCNL	read a RAPID channel
RDPRSN	read a single person from a tenability history file
RDTENA	read a TENAB channel
READAS	read from IOFILI a single entry, in ASCII format
READCF	read the configuration file for OPENSHEL
READCV	convert data in CFIO to real and fixed format
READIN	read a line of ASCII data and process it (calls readas, convrt,...)
READIT	convert a string into useful interger and floating point numbers
READKB	interface for low level keyboard read, mouse, and filter functions
READMN	allow a negative sign on data (not normally allowed) (plot)
READOP	read options, put into the environment CSHELL
READPLOT	read and sort information to place plots
RESETLUT	reset the color palette for EGA and above
RESTRT	read an interval from a history file
RESULT	write results from the model to IOFILO (model)
RSLT	write results from the history file to IOFILO (report)
RSMS	reset the mouse counters
RUNFILE	save a main data file (editor)
SAVE	save (write) an ASCII file (plot)
SAVRAP	save (write) and ASCII file in the RAPID format (plot)
SETAX	set axes parameters, scale labels, ... (plot)
SETAXIN	get axes parameters manually from the keyboard (plot)
SETCLR	set foreground and background colors (editor)
SETEGA	get the palette for color displays (ega and above) - save in order to restore
SETLUT	set the look up table - null procedure, but necessary for compatiblity
SETUNT	set units
SOLVE	top level of the solver for the predictive equations - calls difeq, hvac,...
SOLVEIT4	top level of the solver for the predictive equations (editor)
SSTRNG	parse a string
SSTRNGP	parse a string, and allow parenthesis as delimiters (plot)
STPORT	species transport
SUNITS	set units

SWINDOW	save (in a buffer) a portion of a screen (plot)
TELLKEYS	tell what the function keys do (editor)
TOPSCR	clear the top portion of a screen (BOTSCR clears the bottom) (editor)
TOUPPER	convert ASCII from lower to upper case
TOXIC	calculate the parts per million of gases, and the concentration time dose
TOXICB	fill a table box (model) for doses and other such stuff
TOXICH	" " " " " " "
TOXICR	" " " " " " "
TTCB	damping for the pressure fluctuations at very low pressure differences
UP_FIRE	go through the calculation of heat of combustion, pyrolysis rate, ...
UP_VENT	calculate the constraints on the vents used for horzontal flow
VWPRT	calculate the window size for a view within a small window
WDDRAW	draw the stroke characters (model)
WHICHONE	command processor (plot)
WR_VENT	write the vent parameters to a buffer for subsequent display
WRITEOT	write one record of a history file

The following routines are not executed by any of the modules, but provide initialization of the named common blocks, and are called block data statements. They must be included in the appropriate **makefile.**

INITBK	initialize the CFIN common block
INITDI	initialize the DISPLAY common block
INITLV	for the labels in the editor in CFIN
INITUN	initialization for the units in CFIN
POSDAT	data for cursor positioning by the routine POSIT
INITCS	shell parameter initialization, such as input/output unit numbers, CSHELL
NAILEDF	zero the fed counters in the tenability data for the report generator
INITPL	plot specification initialization, such as the graphics output device,... in DFLTS, DFLTS1 and PLTFLT

REFERENCES

[1] Forney, G. P. and Cooper, L. Y., The Consolidated Compartment Fire Model (CCFM) Computer Application CCFM.VENTS - Part II: Software Reference Guide, Nat. Inst. Stand. and Tech. Internal Report 90-4343 (1990).

[2] Jones, W. W., A Multicompartment Model for the Spread of Fire, Smoke and Toxic Gases, Fire Safety Journal *9*, 55 (1985); Jones, W. W. and Peacock, R. D., Refinement and Experimental Verification of a Model for Fire Growth and Smoke Transport, Proceedings of the 2nd International Symposium on Fire Safety Science, Tokyo (1989); Jones, W. W. and Peacock, R. D., Technical Reference Guide for FAST Version 18 Nat. Inst. Stand. and Tech. Tech. Note 1262 (1989).

[3] Forney, G. P. and Jones, W. W., Software Development Tools, Nat. Inst. Stand. and Tech. Internal Report 90-4363 (1990).

[4] American National Standard Programming Language Fortran, X3.9-1978 (ISO 1537-1980 (e)), American National Standards Institute, New York, NY (1978).

[5] Polytron Version Control System User's Reference Manual, Sage Software, Beaverton, OR (1989).

[6] F77L Fortran Reference Manual, Lahey Computer Systems Inc., Incline Village, NV (1990)

[7] Plink86+ User Manual, Phoenix Technologies LTD., Norwood, MA (1987)

[8] PolyMake User's Manual, Sage Software, Beaverton, Oregon (1989)

[9] Abramowitz, M and Stegun, I., Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55 (1964).

[10] Jones, W. W. and Fadell, A. B., A Device Independent Graphics Kernel, NBS (U.S.) Internal Report 85-3235 (1985).

NIST-114A
(REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NIST/TN-1283

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
November 1990

4. TITLE AND SUBTITLE

A Programmer's Reference Manual for CFAST, the Unified Model of Fire Growth and Smoke Transport

5. AUTHOR(S)

Walter W. Jones and Glenn P. Forney

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED
Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

Same as item #6

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

This document describes the unified model of fire growth and smoke spread, CFAST. This paper documents the internal structure of the model and details the method of modifying the model, together with examples. The intent is to provide a framework and methodology for maintenance of the model, together with a method of updating it. The reader is assumed to have a working knowledge of programming, software maintenance and modeling of physical phenomena.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

compartment fires; fire growth; mathematical models; numerical models; software development

13. AVAILABILITY

X	UNLIMITED
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
X	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
X	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
104

15. PRICE

ELECTRONIC FORM

☆ U.S. GOVERNMENT PRINTING OFFICE: 1990 — 281-557/20743

NIST Technical Publications

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW., Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NIST research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NIST publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency Reports (NISTIR)—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Institute of Standards and Technology
(formerly National Bureau of Standards)
Gaithersburg, MD 20899