FINAL REPORT
COUPLING THE FIRAC AND CFAST COMPUTER CODES

October 1993

S.W. Claybrook

Numerical Applications, Inc.
825 Goethals Drive, Suite A
Richland, WA 99352

prepared for
Los Alamos National Laboratory

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

CONTENTS

	Page
INTRODUCTION	1
FIRAC DEBUGGING AND ERROR CORRECTIONS	1
FIRE COMPARTMENT INTERFACE GENERALIZATION	1
FIRAC/CFAST COUPLING	2
FIRAC/CFAST PREPROCESSOR	4
FIRAC/CFAST POSTPROCESSOR	4
RECOMMENDATIONS	5

INTRODUCTION

This report summarizes the work performed by Numerical Applications, Inc. on LANL subcontract 3876L0013-9Q. The primary objectives of this work were to generalize the fire compartment interface in the IBM PC version of FIRAC and to couple FIRAC with the CFAST computer code. The resulting FIRAC/CFAST computer code would combine the ventilation system and particulate transport modeling capabilities of FIRAC with the fire room modeling capabilities of CFAST. Additional project objectives included debugging FIRAC to correct errors that had been reported by the FIRAC-PC evaluators and evaluating requirements for modifying the FIRAC preprocessor and postprocessor to work with the combined code.

FIRAC DEBUGGING AND ERROR CORRECTIONS

One of the major FIRAC-PC problems that had been reported was a flow mismatch across a fire room when the FIRIN module was being used. Debugging efforts revealed some differences in the units conversion factors used in FIRAC and FIRIN, which caused an inconsistency between the fire room pressure and the pressures in the rest of the ventilation system. This inconsistency was corrected, but it only partially resolved the flow mismatch problem. The remainder was found to be related to the use of internal boundary nodes in FIRAC to represent a fire room.

Since boundary nodes are used to model the fire room, the pressure at these nodes is set to an initial value and then controlled by the the FIRIN module during a transient calculation. When FIRAC solves for an initial steady state flow condition, it is essentially solving two separate flow problems; one for the ventilation system components on the inlet side of the fire room and the other for the components on the outlet side of the fire room. If there are any inconsistencies in the specified input data for initial flows, pressure drops, and loss coefficients, then these inconsistencies can result in a flow mismatch on either side of a FIRIN room. The sample problem where this flow mismatch was most apparent did contain inconsistencies in the specified input data.

This example points out the necessity of ensuring consistency among specified branch component pressure drops and loss coefficients, initial room pressures, and blower curves. Currently, the FIRAC user must check the input data manually to ensure that is consistent. As a future modification, automatic consistency checks could be added to the FIRAC preprocessor and a warning message displayed if any inconsistencies are found.

A second FIRAC problem that had been reported earlier was described as a memory mapping error. Debugging efforts showed that this problem resulted from mismatched arguments between certain gas species transport subroutine calls and the actual subroutine definitions. These mismatched arguments were corrected and the gas species transport option was tested and verified to be working correctly.

FIRE COMPARTMENT INTERFACE GENERALIZATION

The first step in coupling CFAST with FIRAC was to generalize the fire compartment interface. The original coupling between FIRAC and the FIRIN compartment model was accomplished largely by passing data through COMMON blocks. Although the method worked, it resulted in a very poorly defined interface that could not be applied to another compartment model such as CFAST.

The original FIRAC/FIRIN interface was cleaned up by first separating all global variables into COMMON blocks applicable to either FIRAC or FIRIN, but not both. Only FIRAC COMMON

blocks were retained in FIRAC subroutines and only FIRIN COMMON blocks were retained in FIRIN subroutines. Necessary communications between FIRAC and FIRIN were identified and special interface subroutines were written to perform those communications. Data that needed to be passed between FIRAC and the interface subroutines was confined to subroutine arguments. Selected FIRIN COMMON blocks were included within the interface subroutines to provide access to the required FIRIN variables. A fire module flag, IFIREM, also was added to FIRAC so that multiple fire modules could be accommodated and selected by the user as required.

All of the above modifications provided a well defined method of communication that worked with the FIRIN compartment model and also was general enough that it could be applied to other compartment models. After the FIRAC/FIRIN interface modifications were complete, an appropriate sample problem was run to ensure that FIRAC still produced results that were identical to those obtained prior to the modifications.

The interface modifications described above identified specific FIRAC subroutines that need to be revised whenever a new fire compartment model is to be added. These subroutines and the tasks that need to be performed there are listed below.

SUBROUTINE INPROC: Subroutine INPROC reads all input data required by FIRAC and the fire module that is active for a particular problem. Flags IFIREM and IFIRIN should be set appropriately for the selected fire module and an interface subroutine called to read the fire module input data.

SUBROUTINE MASSGE: Subroutine MASSGE performs any input data manipulations that are required, such as units conversions. An interface subroutine should be called here to perform data manipulations for the selected fire module.

SUBROUTINE TFNS: Subroutine TFNS is where data is passed between FIRAC and the active fire module. The fire module should be called first to advance the fire room time step. At appropriate locations within the fire room code, interface subroutines should be called to transfer ventilation system mass and enthalpy flows to the fire module. Following the fire module calculations, additional interface routines should be called to update the fire room node temperatures, pressures, particulate concentrations and gas species concentrations in FIRAC.

SUBROUTINE PTRANS: Subroutine PTRANS sets internal boundary node particulate concentrations before solving the particulate species transport equations. The fire module flags should be checked to ensure that this is done whenever a fire module is active.

SUBROUTINE GTRANS: Subroutine GTRANS sets internal boundary node gas concentrations before solving the gas species transport equations. The fire module flags should be checked to ensure that this is done whenever a fire module is active.

SUBROUTINE OUTFLE: Subroutine OUTFLE writes output data to the FIRAC graphics file. The fire module flags should be checked and output data written accordingly.

FIRAC/CFAST COUPLING

After generalizing the fire module interface as described above, the CFAST fire model was coupled to FIRAC. Several new interface routines were written to perform the necessary communications between the two codes. These interface routines and the tasks they perform are described below.

SUBROUTINE CFSINP: Subroutine CFSINP reads the necessary FIRAC/CFast interface data and then calls subroutine CFast to read the input required for the CFast module. CFSINP is called from FIRAC subroutine INPROC.

SUBROUTINE CFSMSG: Subroutine CFSMSG performs units conversions for FIRAC/CFast interface duct diameters and elevations. CFSMSG is called from FIRAC subroutine MASSGE.

SUBROUTINE CFSTME: Subroutine CFSTME transfers total mass and enthalpy from the FIRAC ventilation system to the CFast fire room. CFSTME is called from CFast subroutine DSOURC.

SUBROUTINE CFSTSP: Subroutine CFSTSP transfers individual gas and particulate species masses from the FIRAC ventilation system to the CFast fire room. CFSTSP is called from CFast subroutine STPORT.

SUBROUTINE CFSTMP: Subroutine CFSTMP transfers fire room temperatures from CFast to the FIRAC internal boundary nodes that represent the fire room. CFSTMP is called from FIRAC subroutine TFNS.

SUBROUTINE CFSPRS: Subroutine CFSPRS transfers fire room pressures from CFast to the FIRAC internal boundary nodes that represent the fire room. CFSPRS is called from FIRAC subroutine TFNS.

SUBROUTINE CFSYWQ: Subroutine CFSYWQ transfers fire room particulate species (soot) concentrations from CFast to the fire room internal boundary nodes in FIRAC. CFSYWQ is called from FIRAC subroutine TFNS.

SUBROUTINE CFSGCH: Subroutine CFSGCH transfers fire room gas species concentrations from CFast to the internal boundary nodes that represent the fire room in FIRAC. CFSGCH is called from FIRAC subroutine TFNS.

Several modifications to the CFast source code provided by NIST also were required to couple FIRAC and CFast. Those CFast subroutines that were changed are listed below.

SUBROUTINE CFast: Subroutine CFast, which was changed from the main program to a subroutine, reads the CFast input. It was modified so the CFast input file name could be passed to it from FIRAC. Selected operations that are no longer required were eliminated. Subroutine CFast is called from interface subroutine CFSINP.

SUBROUTINE CFSOLVE: Subroutine CFSOLVE (formerly subroutine SOLVE) was renamed to avoid a conflict with a FIRAC subroutine by the same name. Modifications were made so the time and time step size, which are now controlled from within FIRAC, could be passed to CFSOLVE as subroutine arguments. Calls to the CFast mechanical ventilation subroutines also were deleted. CFSOLVE is called from FIRAC subroutine TFNS.

SUBROUTINE DSOURC: Subroutine DSOURC was modified to get total mass and energy contributions from the FIRAC ventilation system. This was done by calling interface subroutine CFSTME.

SUBROUTINE STPORT: Subroutine STPORT was modified to get individual species mass contributions from the FIRAC ventilation system. This was accomplished by calling interface subroutine CFSTSP.

SUBROUTINE READOP: A minor modification was made to subroutine READOP to accommodate a compiler difference related to getting the system date.

SUBROUTINES DUMPER and ADUMPER: Both of these subroutines required minor changes to OPEN statements to accommodate compiler differences.

In the process of interfacing FIRAC and CFAST, each FIRAC function and subroutine was placed in a separate source file. A 'make' file, fcfast.mak, was created to compile the combined code into a single executable called fcfast.exe. This is consistent with the method used previously to compile CFAST. FIRAC COMMON blocks also were placed in separate header files that could be accessed with INCLUDE statements. Parameter statements were located in the header file fcp.h and used to define the FIRAC array dimensions. This will allow the FIRAC arrays to be redimensioned more easily if it is necessary in the future.

After the FIRAC/CFAST interface modifications were complete, an appropriate sample problem was prepared and run to verify that the CFAST module operated as intended and that the calculated results were reasonable.

FIRAC/CFAST PREPROCESSOR

The FIRAC preprocessor was evaluated to determine those modifications that would be necessary to make it work with the combined FIRAC/CFAST code. It was decided that the most efficient method of providing a preprocessor for FIRAC/CFAST would be to combine the existing FIRAC (FP) and CFAST (CEDIT) preprocessors.

The FIRAC preprocessor was modified to include CFAST room components that could be connected to the FIRAC ventilation system. Provisions were made to obtain the necessary interface data from the user and to allow the user to access CEDIT to prepare the CFAST related input. Annotation and drawing capabilities also were added so that any CFAST rooms that are not connected to the FIRAC ventilation system could still be included in the preprocessor picture. All new features of the combined preprocessor included help windows to assist the user in learning to use the preprocessor. The executable file for the new FIRAC/CFAST preprocessor is called fcp.exe.

FIRAC/CFAST POSTPROCESSOR

The FIRAC and CFAST postprocessors were evaluated to determine the most efficient means of providing plotting capabilities for FIRAC/CFAST. The FIRAC postprocessor, POST, can be used as is to obtain available FIRAC ventilation system data from the graphics file. However, certain data that is of interest is currently not written to the FIRAC graphics file. Both FIRAC/CFAST and POST should be modified to make additional ventilation system data available for plotting.

The current CFAST postprocessor, CPLOT, cannot access the graphics data file produced by FIRAC/CFAST because of differences in the way the SVS and LAHEY compilers handle unformatted files. This problem can be resolved by recompiling CPLOT with SVS FORTRAN, which was used to compile both FIRAC/CFAST and POST. CPLOT contains numerous

graphics calls, which are not standardized among different compiler vendors. Therefore, a substantial amount of effort will be required to convert CPLOT to SVS FORTRAN.

RECOMMENDATIONS

Although the FIRAC/CFAST code is usable in its current form, it should not be distributed to the general fire protection community until the plotting capabilities are fully operational and complete documentation is available. Consideration also should be given to adding other code capabilities to increase its usefulness. Consequently, the following additional tasks are recommended.

1. Prepare a comprehensive FIRAC/CFAST user's manual. The manual should include instructions for using all of the modeling features of the FIRAC/CFAST code as well as its related pre- and postprocessing utilities. Installation instructions and sample problems also should be included in the manual. Manuals for the separate FIRAC and CFAST codes and related processors currently exist which could be modified and combined to create a comprehensive manual for the new code system.
2. Add data checking and diagnostic features to the FIRAC/CFAST preprocessor. The current preprocessor version provides a convenient, graphically oriented interface for entering all input data. However, it is still up to the modeler to ensure that the data is complete and consistent. Modifications could be added to the preprocessor to check input data for completeness and consistency and to display diagnostic messages as appropriate. This would be extremely useful, particularly to less experienced modelers or to those who use the code only occasionally.
3. Prepare a comprehensive postprocessor for FIRAC/CFAST. As a minimum, this involves converting CPLOT to SVS FORTRAN and adding additional ventilation data to those currently available with POST. An additional enhancement that may be considered is to integrate the plotting capabilities into the FIRAC/CFAST preprocessor. This would allow the model picture to be used to graphically select those locations for which data plots are desired.
4. Add radioactive species generation and transport to the FIRAC/CFAST code. CFAST currently considers smoke and gaseous species production from fires, but airborne radioactive particulates are not modeled.
5. Perform comparisons of FIRAC/CFAST results with available fire test data. To date, only minimal code testing has been performed to ensure individual code modules are working as intended and results from FIRAC/CFAST look reasonable. Comparisons with test data should be performed to provide additional confidence in calculated results.