ABSTRACT

Title of Thesis: 'The Conversion of the Process Flowsheet Simulator, FLOWTRAN, For FORTRAN 66 Based Mainframe Computers.'

Timothy E. Roche, Master of Science in Chemical Engineering

Thesis directed by: Professor Edward C. Roche, Jr., Sc.D., P.E.

The process flowsheet simulator FLOWTRAN was converted from FORTRAN 77 code to FORTRAN 66 code. The conversion was accomplished by developing a number of subprograms to handle FORTRAN 77 intrinsic functions that are missing in the FORTRAN 66 standard and by altering the code to take advantage of the developed subprograms. Also developed were a set of procedure files for executing the FORTRAN 66 FLOWTRAN on a Sperry-Univac 90/80-4. The new FLOWTRAN was tested and performs identically to the FORTRAN 77 version. The FORTRAN 66 FLOWTRAN can be installed on any mainframe computer with a FORTRAN 66 compiler.

The Conversion of the Process Flowsheet Simulator, FLOWTRAN, For FORTRAN 66 Based Mainframe Computers

by

Timothy Edward Roche

Thesis submitted to the Faculty of the Graduate School of the New Jersey Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Chemical Engineering
1986

APPROVAL SHEET

Title of Thesis: The Conversion of the Process Flowsheet Simulator, FLOWTRAN, For FORTRAN 66 Based Mainframe Computers

Name of Candidate: Timothy Edward Roche
Masters of Science in Chemical Engineering

Thesis and
Abstract Approved: ______________________________ 4/28/86
Edward C. Roche, Jr., ScD — Date
Professor
Department of Chemical Engineering

______________________________ 4/29/86
Date

______________________________ 4/29/86
Date

VITA

Name: Timothy Edward Roche

Permanent Address:

Degree and Date to be conferred: M.S.Ch.E., 1986

Date of Birth:

Place of Birth:

Secondary education: Parsippany Hills High School, 1980

Collegiate institutions attended	Dates	Degree	Date of Degree
New Jersey Institute of Technology	80-84	B.S.Ch.E.	1984
New Jersey Institute of Technology	84-86	M.S.Ch.E.	1986

Major: Chemical Engineering

DEDICATION

To My Parents: Nancy I. Roche and Edward C. Roche, Jr.

Acknowledgement

I would like to thank Steve Keeton for his help on the initial stages of the conversion, which eventually lead to the creation of the EQUAL and MYINDX subroutines. And a special thanks to my thesis advisor and father, Dr. Edward C. Roche, Jr. for help throughout the work on this thesis, and particularly on the procedure files.

TABLES OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I INTRODUCTION

The objective of this thesis was the conversion and implementation of the process simulator FLOWTRAN on a computer equipped with a FORTRAN 66 or FORTRAN IV compiler. The complete process simulator, here after referred to as FLOWTRAN, supplied to the author was written for a FORTRAN 77 compiler. The conversion entailed alterations to the original code so that FLOWTRAN would compile on a FORTRAN 66/FORTRAN IV compiler and then run error free. The purpose for this conversion was to permit the use of FLOWTRAN on computers that do not have a FORTRAN 77 compiler available, for example the Sperry-Univac 90/80-4 which has only the 1966 standard FORTRAN compiler. This FORTRAN 66 version of FLOWTRAN can be made available to other schools that do not have a computer capable of handling the FORTRAN 77 version. The thesis was limited to producing a version of FLOWTRAN that performed as close to the original as possible. This goal was accomplished by creating general subprograms and algorithms that would emulate the inherent capabilities of the FORTRAN 77 code.

II FLOWTRAN

FLOWTRAN is a sequential modular process simulator: a generalized steady state modeling computer program for process design and plant simulation (Seader iii). The code was developed by Monsanto's Applied Mathematics Department during the 1960's. FLOWTRAN allows comprehensive modelling of process flow sheets with relative ease of use. Along with flow sheet simulation FLOWTRAN is capable of retrieving physical data from its library, estimating physical properties from established correlations (including petroleum fractions), and correlating vapor liquid equilibrium data. All of these abilities are handled by the four main programs that make up the FLOWTRAN package.

The four main programs that constitute the FLOWTRAN package are: (1) INF, the information retrieval system; (2) PROPTY, the physical property correlator and estimator; (3) VLE, the vapor liquid equilibrium correlating program; (4) PREPRO, the FLOWTRAN-flow-sheet-simulator interpreter.

The program PREPRO develops from the user supplied input data a FORTRAN main program which performs the actual flow sheet simulation. All four main programs and their accompanying subprograms are written in FORTRAN. Approximately 40,000 lines of FORTRAN code comprise the FLOWTRAN package. The four main programs in the FLOWTRAN package also share similarities in the way they are executed and the input/output files that are used.

The input data files for the four main programs are similarly formatted (examples are included in Appendix E). Each line of input is a record. Lines beginning with a blank are considered continuations of the previous record. Two arguments must be separated by at least one blank space. Argument lists may be continued on additional lines by leaving at least the first column blank on each additional line. While an argument may end in the last column of a line, it cannot be split between lines. The first word on any record is the keyword (unless the record contains a blank in the first column). The keyword defines the type of record and whatever follows within the record. As an example consider the record:

```
BLOCK DIS DSTWU FEED BOTM OVHD
```

The keyword is BLOCK which indicates that a subroutine will be called. BLOCK records contain a label for the block, in this case DIS, short for distillation tower. After the label comes the block-type, DSTWU, which is the distillation block using the Winn-Underwood algorithm (a complete list of blocks is contained in Appendix B). Following the block-type on the BLOCK records are the stream names for the feed stream, the bottoms product stream, and the overhead product stream. Because all four main programs' input data files are structured the same, the scanning procedure used by all the main programs in the FLOWTRAN package is very similar.

The above block statement is translated into the following FORTRAN statement.

```
CALL DSTWU(.......)
```

All four main programs in the FLOWTRAN package scan their data files for keywords in the same way. The process involves reading the first six characters on any given line into a temporary scan variable. The scan variable is tested to see if it is a valid keyword for the particular program scanning it. If the scan variable does not match any of the keywords an error diagnostic is generated. On the other hand, if the scan variable does match a keyword then the program proceeds to properly process the information contained on the record. This involves stripping off further keywords to test for validity and reading numerical values from the remainder of the record. This procedure is repeated until an 'END JOB' record is read.

To make this thesis more understandable a short description of the four main programs is presented.

II.1 INF

The information storage and retrieval system, INF, plays a dual role for FLOWTRAN. INF runs as a stand alone program; usually performing simple tasks such as displaying the constants for a chemical specie or listing the chemicals in the library. INF also serves as an unseen servant for the main programs PROPTY and PREPRO. When either of these programs want to retrieve or store component data records they use INF.

INF has two primary functions: retrieving the records for desired chemicals, and maintaining the library through updates of, additions of, and deletions of data records. INF is the librarian for the FLOWTRAN simulation program.

The data library for FLOWTRAN is composed of two or more sections. The main library section is the public file which contains 180 records of chemicals (see Appendix A for list). The public file is a read only file. Read only status permits data retrieval and copying privileges, but does not permit modification to any of the records. The other section of the library are the private files. A private file is distinguished from the public file because it is created by the user, but still can be maintained through the program INF.

The data records in the Physical Property library, whether the records are in the public or Private files, are structured as shown in Table II.1.1

TABLE II.1.1 Physical Data Record

Basic Properties	Common Symbol	FLOWTRAN Symbol	Units
Molar weight	MW	MW	
Normal boiling point	T_{Nbp}	NBP	^oR
Critical temperature	T_c	TC	^oR
Critical pressure	P_c	PC	psia
Critical compressibility	z_c	ZC	
Liquid volume constant*		VL	
Liquid volume (60^oF and 14.7 psia)		LDEN60	gal/lbmole
Enthalpy Properties			
Liquid Enthalpy constant*		HL	
Ideal gas heat capacity constants	$a_1 \ldots a_5$	CP(i)	$\frac{BTU}{lbmole-^oF}$
Liquid Enthalpy constants*		LH(i)	
Equilibria Properties			
Solubility parameter	δ	DELTA	$(cal/ml)^{1/2}$
Expansion factor*		EXPF	
Acentric factor		OMEGA	
Antoine vapor pressure constants	a_1, a_2, a_3	VPA(i)	psia, ^oF
Cavett vapor pressure constants*	a_1, a_2	VPC(i)	psia, ^oF

* Proprietary Monsanto constants computed by PROPTY

When INF is called upon to retrieve named data records, to edit property constants, or to store new data records it does these tasks in the following manner (ref. Fig. II.1.1). INF scans the commands of the data file for recognized keywords and determines the desired function. For the purpose of illustration lets use copying a data record from the public file to the private file, and then print out the private file version. First INF must search the public file to verify that the record exists. When the record has been found it is copied from the data file to temporary memory.

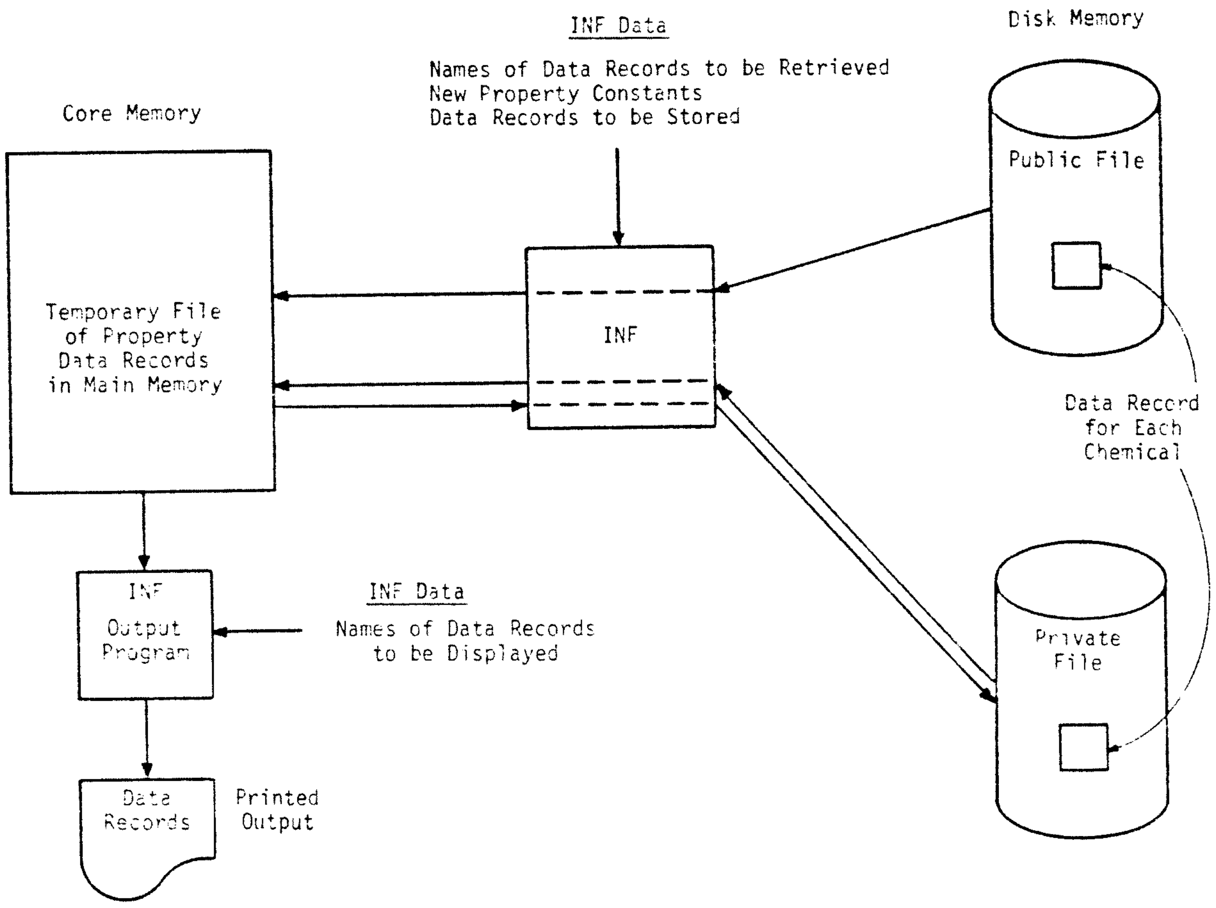
INF Data
Names of Data Records to be Retrieved
New Property Constants
Data Records to be Stored
Disk Memory
Core Memory
Public File
Temporary File of Property Data Records in Main Memory
INF
Data Record for Each Chemical
INF Output Program
INF Data
Names of Data Records to be Displayed
Private File
Data Records
Printed Output

FIGURE 3.1 *Data Storage and Retrieval*

Next, the program will search the private file to find out if the record already is there; if the component name being transferred has no record in the private file it is added to the end of the list; if the component name already exists, the data record is updated by overwriting the previous values. Because an output of the component data was asked for, the information in temporary memory is printed out. A schematic of information flow is presented in Fig. II.1.1 (see facing page).

II.2 PROPTY

When the components for a process are not in the Public data file it becomes necessary to create new component records in the Private data file. This is the major purpose of the program PROPTY. Given raw data as illustrated in Fig. II.2.1, PROPTY generates the liquid enthalpy constants and the expansion factor using proprietary Monsanto correlations.

A PROPTY job runs much like an INF job. First of all, the PROPTY data file is scanned to established what is given and what must be established. Typically a PROPTY job will consist of the information shown in Table II.2.1

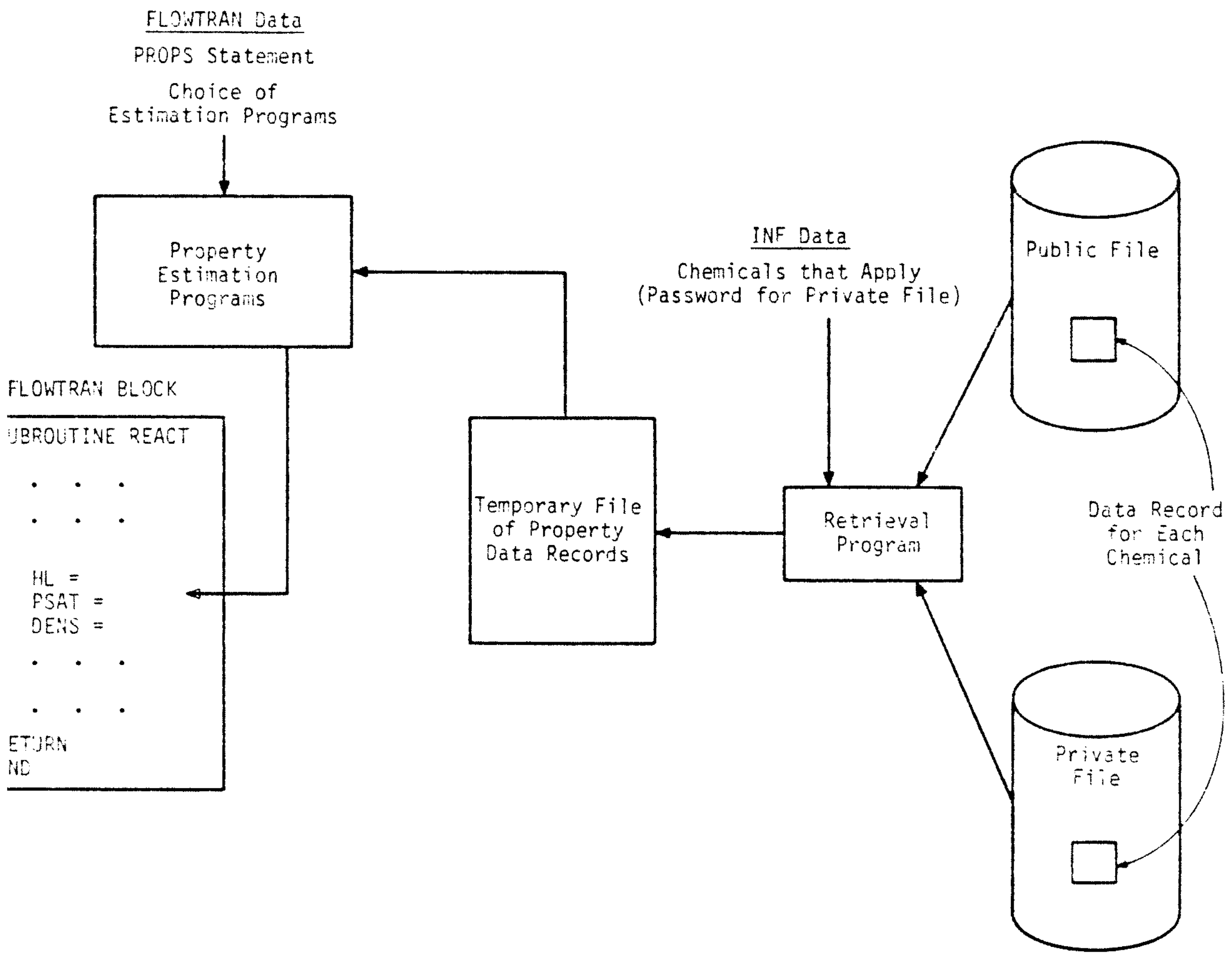
FLOWTRAN Data
PROPS Statement
Choice of Estimation Programs
Property Estimation Programs
INF Data
Chemicals that Apply (Password for Private File)
Public File
FLOWTRAN BLOCK
UBROUTINE REACT
HL =
PSAT =
DENS =
ETURN
ND
Temporary File of Property Data Records
Retrieval Program
Data Record for Each Chemical
Private File

FIGURE 3.4 *Property Estimation*

TABLE II.2.1 Raw Data for Correlation

Component name
Molar weight
Normal boiling point
Critical temperature
Critical pressure

Latent heat of vaporization at specified temperature

Liquid density*

a) Table of liquid density versus temperature
b) Coefficients $\{d_s, \alpha, \beta, \gamma, t_s\}$ for the liquid density polynomial in the International Critical Tables:

$$d_t = d_s = 10^{-3} \alpha(t - t_s) + 10^{-6} \beta(t - t_s) + 10^{9} \gamma(t - t_s)^3$$

Vapor pressure versus temperature table

Heat capacity of ideal gas*

a) Table of heat capacity versus temperature
b) Bond contribution data

* Data in the form of a, b, or both are acceptable.

Most of this information is incorporated directly into a new physical property record, however, some constants have to be evaluated. The curve-fitting program computes the constants that will be in the data record. All the correlated information and physical constants are saved in a temporary file in the main memory. If a STORE statement is included in the data file the information is permanently stored in the Private file by INF. A schematic representation is presented in Fig. II.2.1 (see facing page).

Making physical property data records for components of higher molecular weight can be accomplished by treating them as groups or pseudo components. This is particularly helpful when dealing with petroleum streams. The reader should consult the Introduction to FLOWTRAN by Seader et.al. for further details.

II.3 VLE

VLE calculates the liquid-phase activity coefficient constants. All other thermodynamic quantities are handled by PROPTY. VLE is able to regress parameters for the following cases:

1. Ideal Solutions
2. Regular Solutions
3. Wilson Equation
4. Van Laar Equation
5. Renon Equation (improved by Monsanto)

Given an equation, VLE regresses for the interaction parameters using nonlinear regression techniques with one or more parameters and a choice of five objective functions. The five objective functions are summarized in the Table II-C-1 along with the appropriate data type.

TABLE II.3.1
VLE Objective Functions and Data Types

Objective Function	Code	Data Type T-P-x-y	T-P-x	T-x-x
$\ln(y/x)_i = \ln K_i$	1	N	--	--
$\ln \alpha_{jr}$	2	N - 1	--	--
$\ln(y/x)_r = \ln K_r + \ln\alpha_{jr}$	3	N	--	--
$\sum(y_i - 1)$	4	1	1	--
$\ln(x^I/x^{II})_i = \ln K_{Di}$	5	--	--	N

Dashes indicate an invalid combination. Other entries indicate valid combinations; the value of the entry indicates the number of equations per experimental point for a system of N components. The subscript i represents all components, r represents a reference component (component i), and j represents all components except the reference component.

II.4 PREPRO

The most frequently used and complicated segment of the FLOWTRAN package is the one built around the main program PREPRO. This program is interchangeably referred to as PREPRO (the preprocessor) or just plain FLOWTRAN, because the most important task done by the FLOWTRAN package is the steady state simulation of process flow sheets, which is PREPRO's function.

The process simulator is constructed with a number of integral parts: the physical property data library and information retrieval system (INF), the preprocessor (PREPRO), and a library of BLOCKS or FORTRAN subprograms that simulate various unit operations.

Given a FLOWTRAN-job-data file describing a process: FLOWTRAN will discover which components are involved and recall their physical properties. FLOWTRAN will determine the property estimation techniques desired and use them when needed. FLOWTRAN will perform the unit operations included in the process and perform the evaluation in the order they are input. FLOWTRAN will read off the stream flow rates, temperatures, and pressures. And finally, FLOWTRAN will simulate the process.

A FLOWTRAN-job run relies on two FORTRAN programs (INF and PREPRO) , the host computer's compiler and linker (to generate the specific simulation program), the public and private physical property data files, and the collection of unit operation blocks (algorithms) that FLOWTRAN is capable of simulating (input to the linker).

The procedure begins with the information retrieval system (INF), which scans the input data file for its own keywords. Specifically it looks for the keywords FILE and RETR. FILE optionally informs INF which data library to look in -- the default is the public file. The keyword RETR tells INF which components are in the simulation and to retrieve the physical property data for each of them. The input file is then rewritten with the needed data inserted where the RETR statement had been. The output from INF is then passed to PREPRO as input.

The procedure continues to the second FORTRAN program, the preprocessor, PREPRO. PREPRO interprets the input data file and creates a new FORTRAN program that will ultimately simulate the process described by the input data file. The process is mapped out using the keywords in Table II.4.1 to fully describe the FLOWTRAN flow sheet.

TABLE II.4.1
Arrangement of FLOWTRAN Input Data

Key Word of Statement	
TITLE	
PROPS	
PRINT *	
FILE *	Component Data Statements
RETR	
PAIR *	
NEW BLOCK *	
BLOCK	Unit Operation Statements
PARAM	
MOLES	
POUNDS or LBS	
TEMP	Stream Input Statements
PRESS	
NOFLSH *	
END CASE	
Parametric case data consisting of many of the above statements	
END JOB	

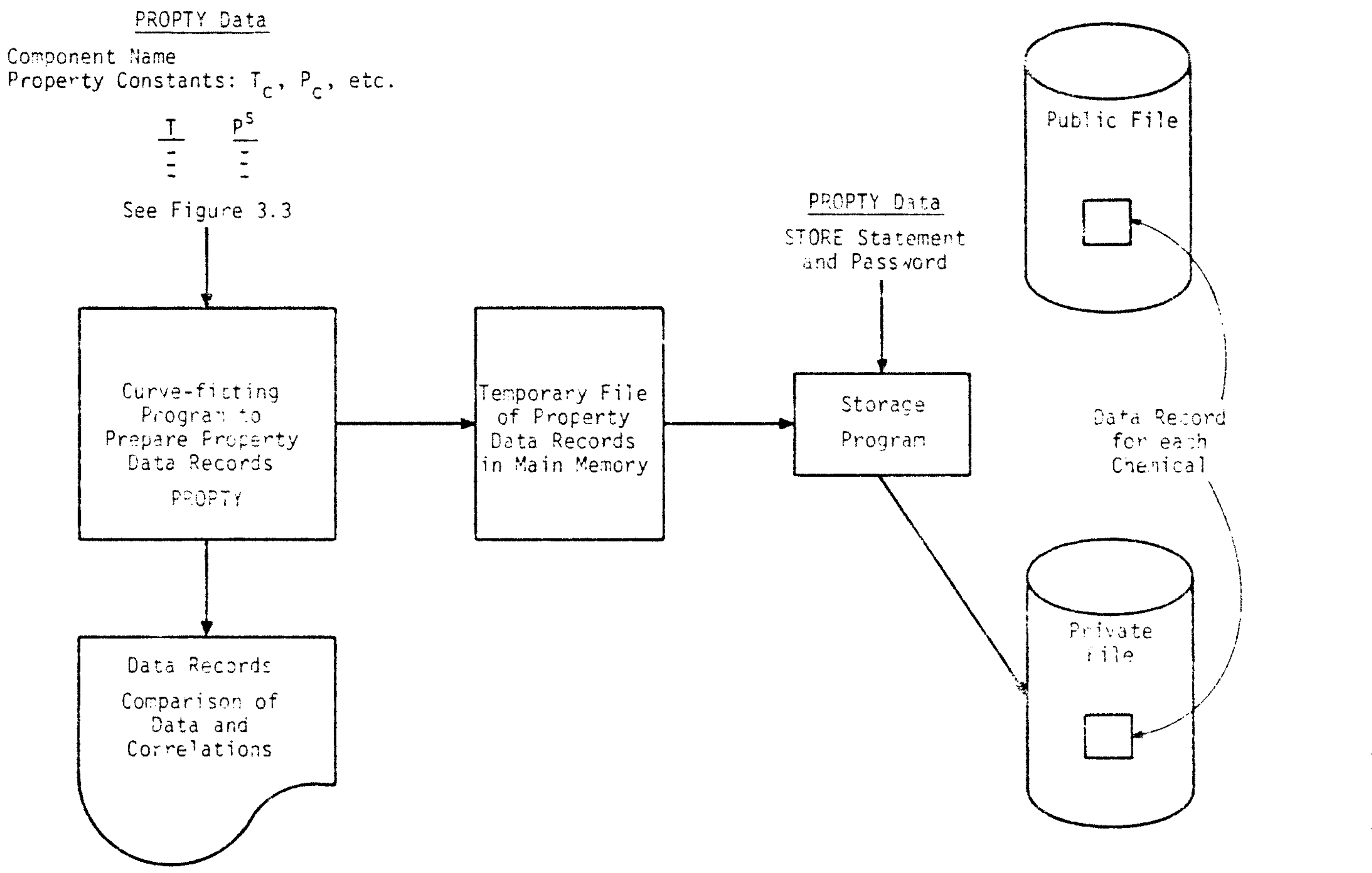
PROPTY Data
Component Name
Property Constants: T_c, P_c, etc.
T
P^s
See Figure 3.3
Curve-fitting Program to Prepare Property Data Records
PROPTY
Temporary File of Property Data Records in Main Memory
PROPTY Data
STORE Statement and Password
Storage Program
Public File
Data Record for each Chemical
Private File
Data Records
Comparison of Data and Correlations

FIGURE 3.2 *Curve Fitting and Data Storage*

* not used in all FLOWTRAN simulations

All the FLOWTRAN keywords have specific meanings to PREPRO but RETR and FILE, which are the keywords used to inform INF to recover the physical property data for the components in the process. The keywords in Table II.4.1 marked Stream Input Statements are self explanatory with the exception of NOFLSH. Normally, FLOWTRAN will perform an equilibrium flash computation to find the phase condition of any feed or tear stream. This calculation presumes only one liquid phase. (Seader 65) To suppress the flash calculation the NOFLSH statement is included.

The remaining keywords bear a little more explanation because they are so important to the set up of the simulation. The first in order of appearance in the data file is the PROPS statement.

The PROPS keyword tells the simulator how many components in the process, and what property estimation options are desired. Their are options on estimation of vapor pressure, vapor fugacity, liquid fugacity, and liquid activity coefficient. An over view of property estimation is presented in Fig. II.4.1 (see facing page).

Next are the FILE keyword and the RETR keyword explained above.

After the RETR statement comes the PAIR record, if it is necessary. PAIR statements are inserted to supply the liquid-phase activity coefficient equation with parameters for binary pairs. (Seader 58)

The NEW BLOCK record signals FLOWTRAN that in addition to the normal blocks, a new block is being used in the simulation. Recognizing the NEW BLOCK statement, FLOWTRAN knows that following the END JOB record is the FORTRAN code for a new block.

The most important of the keywords for FLOWTRAN jobs is the BLOCK statement. The FLOWTRAN process simulator is built around a library of computer programs, each designed to model a basic unit operation in a chemical process plant. Each equation-solving algorithm is represented as a BLOCK statement. The blocks in the FLOWTRAN library are classified defined as follows:

Class	Block Type
1	operation
2	recycle stream convergence
3	feedback controller
4	feedforward controller
5	multi-parameter feedback controller
6	cost and sizing
7	report

A complete list of the available blocks is in Appendix B.

The BLOCK statement serves three principle functions in encoding a FLOWTRAN flow sheet: (1) designate the particular blocks that model the process units, (2) describe how units are connected by information streams and control signals, and (3) indicate the calculation order (the BLOCK statements are arranged in order of computation). (Seader 59)

Each BLOCK statement has an individual parameter statement (PARAM) tied to it. The PARAM statement holds all pertinent information needed for any particular BLOCK statement that is not included in the BLOCK statement.

The END CASE and the END JOB statements are yield and stop signs for FLOWTRAN. END CASE represents the end of one run of the process being simulated under a certain set of parameter values. To run a parametric studies of the FLOWTRAN flow sheet, follow an END CASE statement with PARAMeter statements for any values that need to be changed. The END JOB record signifies the end of the present FLOWTRAN input file. Following this record is the FORTRAN code for NEW BLOCKS.

After translating the data file, PREPRO outputs the main FORTRAN source code file for the simulation. The next steps of the procedure are compilation by the host computer's FORTRAN compiler and linking of the PREPRO prepared main FORTRAN program with the FORTRAN subroutines earlier indicated by the BLOCK statements. The final executable file is the FLOWTRAN simulation of the process.

All that is left to do is run the executable file and print out the results. After execution a clean-up step is carried out, erasing all intermediate files.

III CONVERSION

FLOWTRAN is intimately dependent on the FORTRAN code in which it is programmed in and in which it constructs the main program of any FLOWTRAN job, as described in the previous chapter. A consequence of FLOWTRAN's dependence on the FORTRAN code precludes that it will only execute on a computer that accepts the FORTRAN in which it is written. The majority of computers that are manufactured today have a FORTRAN compiler that executes FLOWTRAN. However, a number of mainframes around the country were built previously to the latest revision of the FORTRAN standard and as a result are unable to use FLOWTRAN. It is for this problem that this thesis addresses itself.

To allow the greatest portability of the FORTRAN language the American National Standards Institute, Inc. set the generic standard on which all compilers should comply. In this way a program written on an IBM computer could as easily be transported to a Control Data Corporation computer and work, or be transported to a Sperry-Univac computer and work. The first standard was published in ANSI X3.9-1966 (FORTRAN 66), the second and current standard was published in ANSI X3.9-1978 (FORTRAN 77). The FLOWTRAN package distributed by CAChE was developed on a computer with a FORTRAN 77 compiler. Mainframes or minicomputers that do not have an available FORTRAN 77 compiler are unable to use

FLOWTRAN. To solve this problem the existing FORTRAN 77 code had to be altered so that ultimately it would compile using FORTRAN 66.

III.1 Differences

Many differences distinguish FORTRAN 66 from FORTRAN 77, but for the purposes of this conversion only certain differences affected the outcome. Of the differences, the most critical were character type variables and any associated functions for altering character strings. Intrinsic functions that are missing in FORTRAN 66 also were a problem.

III.1.1 Type Hollerith

In the old standard FORTRAN 66 character type variables were absent. Instead it had the Hollerith data type, named for Dr. Herman Hollerith, who invented the alphanumeric code in 1889. No variables are declared Hollerith type, they are actually declared as integer or real, but will hold Hollerith type data. When an alphanumeric string is stored into a Hollerith type variable, a Hollerith format is used; typically data strings are put into a Hollerith variable via a DATA statement or via a READ statement.

Formats used for output of Hollerith data and for DATA statements are of the following forms:

$$nHh_1h_2...h_n \quad \text{or} \quad 'h_1h_2...h_n'$$

where n is the length of the string, H denotes the start of the Hollerith string as do the ticks and h is an alphanumeric character; blanks are significant. Because integers or reals must be used to hold Hollerith data, the strings are limited to four characters for integers and reals, unless double precision reals are used; in that case the string could be eight characters long. If longer strings have to be stored they must be broken up and stored in four byte increments in an array (for this conversion the convention of using only integer types to hold Hollerith data was adhered to).

III.1.2 Type CHARACTER

The character handling capabilities of FORTRAN 66 are a serious limitation and FORTRAN 77 took strides to over come these deficiencies. FORTRAN 77 has four types of data: integer, real, logical, and character. The new type character is the focus of this section.

Character data can be represented by constants, variables, or arrays, which in turn may be acted upon by assignment statements, DATA statements, and relational operations.

A character constant is formed by a series of characters enclosed within apostrophes. Characters enclosed within the apostrophes are called a string and are numbered 1, 2, 3, ... consecutively. The characters in a string may contain blanks, but may not be made up of only blanks.

When it is necessary to represent apostrophes within a string they may be included by two apostrophes in a row. The double apostrophes are recognized by the compiler as a single apostrophe because zero length character strings are not allowed.

The length of a string is the number of characters within the apostrophes, with the double apostrophes counting as one character. The delimiting apostrophes are not counted. The string 'TIM''S PROGRAM' has a length of 13, counting the double apostrophe as a single character and including the blank. (Hill 294)

The form of a CHARACTER type statement takes different guises. A variable name may be declared type character by a CHARACTER statement or by an IMPLICIT statement. The length of a character variable is specified in the CHARACTER statement by following the variable name with an asterisk and then an unsigned, nonzero, integer constant that indicates the length. Type-character variable's lengths may also be set up by a default length. A default character variable length may be specified by following the word CHARACTER with the asterisk-integer combination. The length of each entity in the statement that does not have its own length specified will assume the default length.

All characters in an array must have exactly the same length. If a length is not specified for any item, a length of 1 is assigned by default. For example, the length of the character strings for the variable VAR, and of each element of the arrays ARRAY and TABLE, may be established as five by any of the following ways:

```
CHARACTER *5 VAR, ARRAY(6), TABLE(4,3)
CHARACTER VAR*5, ARRAY(6)*5, TABLE(4,3)*5
IMPLICIT CHARACTER*5 (A,T,V)
```

The IMPLICIT statement establishes all entities, not otherwise explicitly defined, that start with the letters A, T, and V as type character with length of five. (Hill 294)

Establishing data in type-character variables may be done by character assignment statements, DATA statements, or by input from a READ statement. Mixing of types is not permitted in either replacement or DATA statements when type character variables are involved. Thus if variables VAR and TABLE have been specified as type character, the following type character assignments are valid:

```
VAR = TABLE(I,J)
TABLE(2,I + 1) = '123'
TABLE(3,I + 2) = 'ABC'
```

On the other hand, the following three statements are not valid.

```
TABLE(2,I + 1) = 123 (Not valid because 123 is type
                        integer.)
TABLE(2,I + 1) = 123. (Not valid because 123. is type real.)
TABLE(2,I + 1) = NONCHR (Not valid if NONCHR has not been
                          specified as type character.)
```

(Hill 295)

Table III.1.1 shows the effect of unequal lengths in the replacement statements, DATA statements, and input/output statements.

Table III.1.1
Treatment of Unequal Lengths for Type CHARACTER

(Note: LEN(v) means the length of variables v (number of characters specified for variable v in an explicit or IMPLICIT type statement).)

Character assignment statement

where type CHARACTER variable v is declared same length as e

if LEN(v) > LEN(e)	v = e + trailing blanks
LEN(v) < LEN(e)	v = e with its rightmost characters deleted

Data statement

DATA v/'constant'/

if LEN(v) > LEN('constant')	v = 'constant' + trailing blanks
LEN(v) < LEN('constant')	v = 'constant' with its rightmost characters deleted

Input Character Constant Into Variable Using Format Aw

if LEN(v) > w of Aw	v = input data + trailing blanks
LEN(v) < w of Aw	v = rightmost characters from within field width w, and with leftmost characters ignored

Output Character Constant From Variable v

if LEN(v) > w of Aw	output = v with its rightmost characters deleted
LEN(v) < w of Aw	output = leading blanks + v

from Structured Programming in FORTRAN by Louis A. Hill, Jr.

Type character variables and constants may be compared with each other in relational expressions. The results will be .TRUE. or .FALSE. depending on the collating sequence. If the two character expressions to be compared are not of the same length, the shorter one will be extended to the right by blanks until it is the same length.

The collating sequence is processor dependent. Most computers use either the American National Standard Code for Information Exchange, ANSI X3.4-1977 (typically referred to as ASCII), or the Extended Binary-Coded-Decimal Interchange Code (typically called EBCDIC). The codes differ from one another on the following way:

```
ASCII
blank ' # $ % + ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; = ? o
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

EBCDIC
blank ? . ( + + $ * ) ; - / , % ' : # o =
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
```

As a consequence of the collating sequence the results of the relational operations .LT., .GT., .LE., and .GT. (less than, greater than, less than or equal, greater than or equal, respectively) will differ when symbols and numbers are included within the strings being compared. FORTRAN 77 has taken steps to alleviate the machine dependence of the relational operators by including the intrinsic functions LGE, LGT, LLE, and LLT (lexically greater than or equal to, greater than, less than or equal to, less than, respectively). These functions insure that ASCII collating sequence is used. Equality and inequality (.EQ., .NE.) are independent of the processor. (HILL 297)

FORTRAN 77 extends type character capabilities by several important ways. FLOWTRAN extensively takes advantage of only one of them (substrings, concatenation, determining length of a string): substrings. However, substrings increase the power of character manipulation by an order of magnitude, because substrings can be combined to form new substrings.

A substring name is formed by following the name of a character variable (or element of a character array) by two optional integer expressions enclosed in parentheses and separated by a colon. The first integer expression specifies the leftmost character position of the substring; the second specifies the rightmost position. If the first expression within the parentheses is missing, the default value is 1, signifying the leftmost character position. If the last expression is missing, the default value is the length of the variable or array element.

Examples summarizing the rules for substrings are shown in Table III.1.2. The material represents only a portion of a computer program and the strings stored on each location are shown (on the right hand side of the table) for each assignment statement. The character string stored in variable L8 is sequentially altered by a series of substring changes taken from element (2,1) of the array AR. (HILL 307)

Table III.1.2
Examples of CHARACTER Substrings

Given:

```
     CHARACTER *3 L3
     CHARACTER *4 L4
     CHARACTER *5 L5, AR(4,2)
     CHARACTER *8 L8, NAM
```

Then:

```
     L3       = NAM(4:6)         yields L3 = '456'
     L4       = NAM( :4)         yields L4 = '1234'
     L5       = NAM(4: )         yields L5 = '45678'
     L8       = NAM( : )         yields L8 = '12345678'
     L8       = NAM              also yields L8 = '12345678'
     L8(2;3) = AR(2,1)(4:5)     yields L8(2:3) = '45'
                                 so that now L8 = '14545678'
     L8(6: ) = AR(2,1)( :3)     yields L8(6:8) = '123'
                                 so that now L8 = '14545123'
```

from <u>Structured Programming in FORTRAN</u> by Louis A. Hill, Jr.

When using substrings in assignment statements, the same positions cannot be shown on both sides of the equal sign. Thus, while

```
A(3:5) = A (7:9)  (Legal)
```

is legal, it is not legal to write

```
A(5:9) = A(8:12)  (Illegal)
```

because characters 8 and 9 are referenced on both sides of the equal sign. The latter could be accomplished with the following two statements.

```
ALT = A(8:12)
A(5:9) = ALT
```

III.2 Solutions

The problems encountered while converting fell under 5 major types: (1) type character assignment statements, (2) type character constants, (3) relational operations involving type character constants and variables, (4) intrinsic function INDEX, and (5) WRITE statements that access internal files. All the problems involve type character in one way or another. In solving these problems the aim was to limit the number of alterations to the original code; this was for two reasons: to make the editing as easy as possible, because there were so many changes to be made through out the 40,000 lines of code, and by limiting the changes to the code, the chance of violating the integrity of it was decreased. Most of the problems have built into them the added problem of dealing with substrings.

III.2.1 Assignment Statements

Type character assignments statements were by far the most common of the problems to be solved. The problem was moving character data previously held in a type character variable (or constant) into another type character variable both of which were now in a type Hollerith variable, which is actually an integer holding type Hollerith data. The original character variables could be of any length -- in FLOWTRAN they ranged from 1 to 448 characters in length. In a worst case, the assignment statement could be equating a substring of one variable to a substring of another variable. This was a worst case because, most likely, the beginnings and endings of the substrings would not match up to the divisions, every four characters, in the INTEGER arrays holding the character strings. Due to this problem and the inherent complexity of it, two subroutines were developed to handle type character assignment statements.

The two subroutines, EQUAL1 and EQUAL4, were written to emulate the FORTRAN 77 assignment statements when used on type character variables.

The first subroutine, EQUAL1, moves a source vector (substring) into a target vector, starting in position one of the target vector. The term vector refers to a substring as it appears in FORTRAN 66; the vector consists of a portion of an integer array, from the position of the first character to the position of the last character. Many times the starting position and the ending position of the vector do not correspond with the breaks in the array elements (every four characters starting from the beginning of the array), and because of this, the vector may include only fractions of the beginning and the end array elements. The target vector is assumed to be at least long enough to receive the source vector. It is often true that the target vector will be a temporary variable added to the code to enable necessary character manipulation and is then used repeatedly in the same module of the code. With this aim in mind, EQUAL1 erases any previously held data in the target before depositing the source vector in it.

The second subroutine, EQUAL4, moves a source vector into a target vector; the starting and the stopping points for both the source and target are not restricted, other than to the established rules for type character assignment statements (see Table III.1.1). When an assignment is made using EQUAL4 string length is checked for consistency. When a source vector is less than the target vector's length, the target vector is filled with trailing blanks. When a source vector's length is greater than the target vector's length, the source vector is cut off at the rightmost character that will fit and then is moved.

The source listings for both EQUAL1 and EQUAL4 are in Appendix C.

Before going into the inner workings of the EQUAL subroutines the protocol for using them should be sketched out. Examples of the use of both EQUAL1 and EQUAL4 are presented in Table III.2.1. In the example the variables LINES and UN were originally type character. The variables ZIP and MATCH were added to overcome other problems. The FORTRAN 77-code lines are commented out and their FORTRAN 66 replacements are underneath them. It can be seen in the example that type character variables larger than 4 characters in length will become integer arrays of one dimension larger than the original. Hence, LINES, a type character array of 21 with a length of 80 becomes a two dimensional integer array 20 by 21. Similarly, UN, a character variable of length 6, becomes an integer array of 2.

Table III.2.1
Examples of Using the Subprograms
EQUAL1 and EQUAL4

```
.
.
.
         COMMON /LINES/ LINES
C        CHARACTER LINES(21)*80
         INTEGER*4 LINES(20,21), ZIP                    (FORT66)
C        CHARACTER UN*6
         INTEGER*4 UN(2), MATCH                         (FORT66)
         DATA ZIP/'0   '/                               (FORT66)
.
.
.
C        UN = LINES(IC)(K1:K2)
         CALL EQUAL4(LINES(1,IC),K1,K2,UN,1,8)          (FORT66)
.
.
.
C        IF (LINES(IC)(K1:K2) .EQ. '0')
         CALL EQUAL1(LINES(1,IC),K1,K2,MATCH)           (FORT66)
         IF (MATCH .EQ. ZIP)                            (FORT66)
.
.
.
```

Previously it was mentioned that EQUAL4 was used when the consistency of length should be checked. In the EQUAL4 example above, the original assignment statement equated the substring LINES(IC)(K1:K2) to the character variable UN. This statement has been replaced by a call to EQUAL4. In the argument list are the name of the source, the starting position of the source vector (substring), the ending position of the target vector, the name of the target, the starting position of the target vector, and the ending position of the target vector. In the example the source name is LINES(1,IC), the 1 signifies that the entire IC vector of the array LINES is to be transferred to EQUAL4. K1 and K2 correspond to the starting and stopping positions of the source vector. The target name is UN and its corresponding starting position and stopping position are 1 and 8, respectively. The stopping position of 8 is used as a fail-safe to assure that there are no extraneous characters in positions 7 and 8. The result of this call is the same as the original assignment statement, which, of course, was the desired result.

Also in Table III.2.1 is an example of how EQUAL1 is typically used. In the example, the original statement is a relational operation between a substring of array LINES and the type character constant '0'. The replacement of this statement requires two FORTRAN 66 statements. The first is the moving of the vector (substring) into a small variable; in this case vector LINES(1,IC) from K1 to K2 is copied into MATCH. MATCH can then be used in the IF statement. The arguments in the CALL statement for EQUAL1 are first the source name, the starting position of the source vector, the ending position of the source vector, and the name of the target. The starting and stopping points in the target are not needed because EQUAL1 erases the entire target from start to finish previous to transfer of the source. This allows for worry free use of the dummy holding variable MATCH for numerous cases within a module of the code.

To describe the inner workings of the EQUAL subprograms EQUAL4 will be used as a model (the FORTRAN code listing is in Appendix C).

The argument list in the SUBROUTINE statement of EQUAL4 follows the basic definition as was sketched out in the protocol for using EQUAL4, previously. The dummy variable names are SOURCE, START1, FINI1, TARGET, START2, FINI2. SOURCE and TARGET, as their names imply, are the source and target integer variable arrays containing type Hollerith data. In the source code, the dimension of 1 for SOURCE and TARGET are a dummy variable convention of FORTRAN 66 that tells the compiler to transfer the dimension from the calling program. START1 and START2 are the starting (leftmost) positions within the integer arrays SOURCE and TARGET, respectively (the positions are determined as if the string was that of a type character variable). Similarly, FINI1 and FINI2 are the finishing (rightmost) positions in SOURCE and TARGET, respectively (when referring to the substring in FORTRAN 66 the term vector will be used).

Table III.2.2
Copying Scheme for EQUAL4

In this example the FORTRAN 77 code had been:

```
     TARGET(11:15) = SOURCE(16:21)
```

The corresponding FORTRAN 66 replacement code is:

```
     CALL EQUAL4(SOURCE,16,21,TARGET,11,15)
```

The original contents of SOURCE and TARGET are

```
     DATA SOURCE/'TO B','E CO','PIED','BY E','QUAL','4   '/
     DATA TARGET/'COPI','ED B','Y   ','    '/
```

```
                                         1           2
                              1234|5678|9012|3456|7890|1234
     SOURCE (INTEGER*4 array) TO B|E CO|PIED|BY E|QUAL|4
     COPY1 (INTEGER*4 array)  BY E|QUAL|4
     IMAGE1 (LOGICAL*1 array) BY EQUAL4
     IMAGE2 (LOGICAL*1 array) Y EQUAL
     COPY2 (INTEGER*4 array)  Y EQ|UAL |
     TARGET (INTEGER*4 array) COPI|ED B|Y EQ|UAL |
```

Given the information on the argument list, EQUAL4 can perform substring equivalences just like the FORTRAN 77 equivalence statement. This was accomplished by a 5 step copying procedure (see Table III.2.2). The first step makes an exact copy into the integer array, COPY1, of the source vector from the array element that contains the starting position of the vector to the array element that contains the finishing position of the vector. Depending on the relative alignment of the vector to the array element breaks the starting position can be in either the first, second, third, or fourth position of array element 1 of COPY1. This relative alignment would be fine for coping if the target's relative alignment was the same. But, more often than not, this is not the case. Consequently, a method of transferring one character at a time was needed.

The one-character transfer dilemma was solved by using a type logical*1 variable, which ordinarily would hold .TRUE. or .FALSE., but here is used to hold a single type Hollerith character. The vector held in the integer array COPY1 is put into LOGICAL*1 array IMAGE1 via an EQUIVALENCE statement. The transfer of the source vector from INTEGER*4 to LOGICAL*1 makes up step 2.

Step 3 is the coping of the original data in the target, so that any positions not overwritten will be consistent with the original. The integer elements of array TARGET are copied into integer array COPY2. COPY2 is equivalenced to logical array IMAGE2.

Step 4 is the most involved, the vector sitting in IMAGE1 is copied character by character into IMAGE2, simultaneously taking into account the offset of the starting position in IMAGE1 and the starting position on IMAGE2, and the length of the vector. The starting positions for IMAGE1 and IMAGE2 and their relative offsets from each other are handled in the subroutine.

After moving the vector characters into IMAGE2 -- which is equivalenced to COPY2 -- the last step is to transfer the correctly aligned vector into TARGET.

EQUAL1 is a variation on the theme of EQUAL4. Rather than coping and making allowances for the starting and finishing positions in the target, EQUAL1 always transfers the vector into the target starting in TARGET's position 1. It is assumed that any vector transferred with EQUAL1 will fit within the designated target. Most importantly, the target is blanked out before each run of the subroutine. This feature allows the use of one dummy variable that may be used repeatedly for solving problems in any given module.

III.2.2 Type Character Constants

Type character constants were used extensively throughout FLOWTRAN, but especially in PREPRO. This type of problem was typically solved by creating a 'constant variable,' which would be defined at the beginning of the subprogram with a DATA statement. There is an example in Table III.2.1. In the example the character constant '0' is used in an IF statement. The remedy was to create a 'constant variable,' ZIP, which is defined by a DATA statement. The 'constant variable' ZIP is then used in lieu of the character constant.

III.2.3 Relational Operations

Once again refer to Table III.2.1 for the original FORTRAN 77 code and the typical solution method. Most often the relational operations involve checking a substring value. To achieve the same effect in FORTRAN 66, the substring is transferred into a holding variable via EQUAL1, which would then be use in the new FORTRAN 66 IF statement. In the example the substring LINES(1,IC) from K1 to K2 is put into the holding variable MATCH. MATCH is then used in the relational operator. The use of EQUAL1 was specifically designed for use in this situation, because EQUAL1 erases any previous data in MATCH before loading the current vector it can be used over and over again within the same subprogram.

III.2.4 Intrinsic Function INDEX

In the FORTRAN 77 version of FLOWTRAN the code makes use of the intrinsic function INDEX in order to provide varying formats for numbers that could have a wide range of values. Briefly, the number would be checked for its relative magnitude, and then using the INDEX function along with some conditional tests chooses the appropriate format for printing out the number.

The INDEX function searches for a substring C2 (see program listing in Appendix C) in a specified character string C1, and if it finds the substring, returns the substrings (C2) starting position in C1. If C2 occurs more than once in C1, the starting position of the first occurrence (leftmost) is returned. If C2 does not occur in C1 the value of zero is returned.

The original function INDEX could take any length substring C2 less than or equal to C1. In FLOWTRAN the substring C2 was always one character in length. Therefore, the function MYINDX is a scaled down version that only works on C2 character substrings of length one. Also needed in MYINDX's argument list is the value of IPASS, which indicates the length of C1.

III.2.5 Internal Files

Internal files are type CHARACTER variables, array elements, arrays, or substrings contained within the program itself. They are used extensively when rescanning input characters after they have been stored in an internal file when changing types.

For the conversion to FORTRAN 66 there are two problems with internal files: there are no type character variables to serve as internal files and there are no internal file READ or WRITE statements allowed in FORTRAN 66.

The solution was to use READs and WRITEs to core, that is, reading and writing to a unit unit assigned to core memory to make the conversions. A typical READ from an internal file

```
      CHARACTER INTERNAL_FILE*60
      REAL ARRAY(4)
      READ (INTERNAL_FILE,'(4F15.0)') (ARRAY(I),I=1,4)
```

would be replaced by the following statements (assume that unit 3 is assigned to core memory):

```
      INTEGER INTERNAL_FILE(15)
      REAL ARRAY(4)
      REWIND(3)
      WRITE(3,10) (INTERNAL_FILE(I),I=1,15)
      REWIND(3)
      READ (3,20) (ARRAY(I),I=1,4)
 10   FORMAT(15A4)
```

```
     20    FORMAT(4F15.0)
```

the rewind statements are necessary to reposition the pointer.

IV TESTING

The testing of the FLOWTRAN programs was two fold. The first test was just to get the programs to compile. The second was to run the test programs supplied to the author by CAChE along with the original code. The test programs numbered over 50 with programs from the text FLOWTRAN Simulation -- An Introduction by Seader, et.al.. In the end all the test cases reproduced the output of the original FORTRAN 77 code.

In Appendix F are a number of examples of FLOWTRAN problems with their output. The first example in particular is unique because the intermediate files have been included so that the steps for simulating a process with FLOWTRAN may be easily followed.

The one example includes the FLOWTRAN job data file (data_file_name.DAT), the FTCI.DAT FORTRAN file, the FTSI.DAT data file, and the FLOWTRAN output file (data_file_name.FTO). The FLOWTRAN job data file is the user supplied data, the original input for FLOWTRAN. INF reads the file looking for the RETR record. Based on the RETR record, INF retrieves the physical property data and inserts it into the original data file, replacing the RETR record. This new file (FTPI.DAT) then becomes the input for PREPRO. PREPRO is then executed. PREPRO generates two output files. The first output file from PREPRO is

FTCI.DAT, the FORTRAN file, responsible for the actual simulation after being compiled and linked with the appropriate subroutines (BLOCKS). The second output file create by PREPRO is the input data file (FTSI.DAT) for the simulation. The last file created in process of simulating the flow sheet is the final output or data_file_name.FTO file (for FLOWTRAN Output file).

V PROCEDURE FILES

In order to run FLOWTRAN, procedure or command files are needed to instruct the computer on what to do. Appendix D contains the procedure files for the Sperry-Univac 90/80-4. There are separate files for the four main programs, INF, PROPTY, VLE, and FT. In addition Appendix D contains the necessary job control language to execute the FORTRAN program FTPRI.FOR, which creates the Private Data Library file with the password MONSANTO.

The procedure files written for the Sperry-Univac 90/80 can be invoked by the following three line driver file (this example specifically for FT, but all the drivers are included in Appendix D):

```
/LOGON
/DO FT.PROC, (FT.TEST01)
/LOGOFF
```

The procedure files handle a variety of tasks. For FT as an example, the procedure file takes the input file name and appends .DAT on it and then retrieves the file. Having a job data file, the procedure file then defines the unit numbers to the the correct file names. An example would be the physical property data files, private and public, which are assigned to the units 18 and 19, respectively. Being set to go, the procedure file then instructs the computer to execute INF. INF runs and creates an output. The procedure file redirects the INF output as input to PREPRO. It then executes PREPRO. PREPRO generates a FORTRAN file, which the procedure file instructs the computer to compile and subsequently link with the required FLOWTRAN BLOCKs (subroutines). Upon completing the compilation and link editing, the last executable file remains: the FLOWTRAN simulation program (FTCI.EXE). The simulation is run and the final output is created. Finally, the procedure file has only to eliminate the intermediate or scratch files that were generated along the way, after which the simulation is completed.

The generation of the stand-alone programs of VLE, INF and PREPRO are accomplished using the generic job control language (JCL) for the program PGM (PGM.EXE) contained in the file PGM.LOAD. The FLOWTRAN Subprogram Library is generated with the JCL contained in the file FT.LOAD.LIBRARY.

The TIME and DATE functions. These are site specific routines (if they exist) for accessing the system clock. Included in Appendix E are the programs developed for a generic FORTRAN based system without specific site/machine access.

When FLOWTRAN is first brought up on a system the public and private physical property files have to be generated. The files needed to do this are contained in Appendix G.

APPENDIX A

Chemicals in the Public Data File
Taken from FLOWTRAN Simulation -- An Introduction
by Seader, et.al.

Empirical Formula	Component Name	Alias
Inorganic Chemicals		
Ar	ARGON	A
Br_2	BROMINE	BR2
CCL_4	CARBON TET	CCL4
CO	*CARBON MONOXIDE	*CO
CO	CARBON MONOXIDE	*CO
$COCL_2$	PHOSGENE	COCL2
CO_2	$CARBON DIOXIDE	$CO2
CO_2	*CARBON DIOXIDE	*CO2
CO_2	CARBON DIOXIDE	CO2
CS_2	CARBON DISULFIDE	CS2
C_2OCL_4	TRICHLOROACETYL-CL	
CLH	HYDROGEN CHLORIDE	HCL
CL_2	CHLORINE	CL2
HI	HYDROGEN IODIDE	HI
H_2	*HYDROGEN	*H2
H_2	HYDROGEN	H2
H_2O	WATER	H2O
H_2S	*HYDROGEN SULFIDE	*H2S
H_2S	HYDROGEN SULFIDE	H2S
H_3N	AMMONIA	NH3
Ne	NEON	NE
NO	NITRIC OXIDE	NO
NO_2	NITROGEN DIOXIDE	NO2
O_2	OXYGEN	O2
O_2S	*SULFUR DIOXIDE	*SO2
O_2S	SULFUR DIOXIDE	SO2
O_3S	SULFUR TRIOXIDE	SO3
Organic Chemicals		
$CHCl_3$	CHLOROFORM	CHCL3
CHN	HYDROGEN CYANIDE	HCN
CH_2O	FORMALDEHYDE	HCHO
CH_3Cl	METHYL CHLORIDE	MECL
CH_3I	METHYL IODIDE	CH3I
CH_4	$METHANE	$C1
CH_4	*METHANE	*C1
CH_4	METHANE	C1
CH_4O	METHANOL	MEOH
CH_5N	METHYLAMINE	
C_2HCl_3	TRICHLOROETHYLENE	TCE
C_2HCL_3O	DICHLOROACETYL-CL	
C_2H_2	ACETYLENE	C2H2

$C_2H_2Cl_2O$	CHLOROACETYL-CL	
C_2H_3Cl	VINYL CHLORIDE	
C_2H_3ClO	ACETYL CHLORIDE	
$C_2H_3Cl_3$	112TRI-CL-ETHANE	CL3C2
C_2H_3N	ACETONITRILE	
C_2H_4	*ETHYLENE	*C2=
C_2H_4	ETHYLENE	C2=
$C_2H_4Cl_2$	1,1-DICHLOROETHANE	11DCE
$C_2H_4Cl_2$	1,2-DICHLOROETHANE	12DCE
C_2H_4O	ACETALDEHYDE	
C_2H_4O	ETHYLENE OXIDE	EO
$C_2H_4O_2$	ACETIC ACID	HOAC
$C_2H_4O_2$	METHYL FORMATE	
C_2H_5Cl	ETHYL CHLORIDE	ETCL
C_2H_6	$ETHANE	$C2
C_2H_6	*ETHANE	*C2
C_2H_6	ETHANE	C2
C_2H_6O	DIMETHYL ETHER	DME
C_2H_6O	ETHANOL	ETOH
$C_2H_6O_2$	ETHYLENE GLYCOL	EG
C_2H_6S	DIMETHYL SULFIDE	DMS
C_2H_6S	ETHYL MERCAPTAN	ETSH
C_2H_7N	ETHYLAMINE	
C_3H_3N	ACRYLONITRILE	ACN
C_3H_4	METHYLACETYLENE	
C_3H_4	*PROPADIENE	*PD
C_3H_4	PROPADIENE	PD
C_3H_6	*PROPYLENE	*C3=
C_3H_6	PROPYLENE	C3=
C_3H_6O	ACETONE	
$C_3H_6O_2$	ETHYL FORMATE	
$C_3H_6O_2$	METHYL ACETATE	
$C_3H_6O_2$	PROPIONIC ACID	C3ACID
C_3H_7NO	DIMETHYLFORMAMIDE	DMF
C_3H_8	$PROPANE	$C3
C_3H_8	*PROPANE	*C3
C_3H_8	PROPANE	C3
C_3H_8O	ISO-PROPANOL	IC3OH
C_3H_8O	N-PROPANOL	NC3OH
C_3H_9N	TRIMETHYLAMINE	TMA
C_4H_4	VINYLACETYLENE	
C_4H_4S	THIOPHENE	
C_4H_5N	METHACRYLONITRILE	MACN
C_4H_6	BUTADIENES	BDS
C_4H_6	DIMETHYLACETYLENE	
C_4H_6	ETHYLACETYLENE	
C_4H_6	*1,2-BUTADIENE	*12BD
C_4H_6	1,2-BUTADIENE	12BD
C_4H_6	*1,3-BUTADIENE	*13BD
C_4H_6	1,3-BUTADIENE	13BD
C_4H_8	BUTYLENES	C4=S

C_4H_8	*1-BUTENE	*1C4=
C_4H_8	1-BUTENE	1C4=
C_4H_8	*CIS-BUTENE	*CSC4=
C_4H_8	CIS-BUTENE	CSC4=
C_4H_8	*ISO-BUTENE	*IC4=
C_4H_8	ISO-BUTENE	IC4=
C_4H_8	*TRANS-2-BUTENE	*TRC4=
C_4H_8	TRANS-2-BUTENE	TRC4=
C_4H_8O	ISO-BUTYRALDEHYDE	IC4HO
C_4H_8O	MEK	
$C_4H_8O_2$	N-BUTRIC ACID	
$C_4H_8O_2$	ETHYL ACETATE	
$C_4H_8O_2$	METHYL PROPIONATE	
$C_4H_8O_2$	PROPYL FORMATE	
C_4H_9NO	DIMETHYL ACETAMIDE	DMA
C_4H_{10}	$ISO-BUTANE	$IC4
C_4H_{10}	*ISO-BUTANE	*IC4
C_4H_{10}	ISO-BUTANE	IC4
C_4H_{10}	$N-BUTANE	$NC4
C_4H_{10}	*N-BUTANE	*NC4
C_4H_{10}	N-BUTANE	NC4
$C_4H_{10}O$	ISO-BUTANOL	IC4OH
$C_4H_{10}O$	N-BUTANOL	NC4OH
$C_4H_{10}O$	T-BUTYL ALCOHOL	TC4OH
$C_4H_{10}O$	DIETHYL ETHER	DEE
$C_4H_{10}O_3$	DIETHYLENE GLYCOL	DEG
$C_5H_4O_2$	FUFURAL	
C_5H_8	ISOPRENE AND C5	ISOPRE
C_5H_{10}	*2-ME-1-BUTENE	*2M1B=
C_5H_{10}	2-ME-1-BUTENE	2M1C4=
C_5H_{10}	*2-ME-2-BUTENE	*2M2B=
C_5H_{10}	2-ME-2-BUTENE	2M2C4=
C_5H_{10}	*3-ME-1-BUTENE	*3M1B=
C_5H_{10}	3-ME-1-BUTENE	3M1C4=
C_5H_{10}	*CYCLOPENTANE	*CP
C_5H_{10}	CYCLOPENTANE	CP
C_5H_{10}	*1-PENTENE	*1C5=
C_5H_{10}	1-PENTENE	1C5=
C_5H_{10}	*CIS-2-PENTENE	*CSC5=
C_5H_{10}	CIS-2-PENTENE	CISC5=
C_5H_{10}	*TRANS-2-PENTENE	*TRC5=
C_5H_{10}	TRANS-2-PENTENE	TRNC5=
$C_5H_{10}O$	DIETHYL KETONE	DEK
$C_5H_{10}O_2$	N-PROPYL ACETATE	
C_5H_{12}	$ISO-PENTANE	$IC5
C_5H_{12}	*ISO-PENTANE	*IC5
C_5H_{12}	ISO-PENTANE	IC5
C_5H_{12}	$N-PENTANE	$NC5
C_5H_{12}	*N-PENTANE	*NC5
C_5H_{12}	N-PENTANE	NC5
C_5H_{12}	*NEO-PENTANE	*NEOC5

C_5H_{12}	NEO-PENTANE	NEOC5
$C_6H_3Cl_3$	1,2,4-TRI-CL-BZ	124TCB
$C_6H_4Cl_2$	M-DICHLOROBENZENE	MDCB
$C_6H_4Cl_2$	O-DICHLOROBENZENE	ODCB
$C_6H_4Cl_2$	P-DICHLOROBENZENE	PDCB
C_6H_5Br	BROMOBENZENE	BRBZ
C_6H_5Cl	CHLOROBEZENE	CLBZ
C_6H_5I	IODOBENZENE	IBZ
C_6H_6	*BENZENE	*BZ
C_6H_6	BENZENE	BZ
C_6H_6O	PHENOL	
C_6H_7N	ANILINE	
C_6H_{12}	*CYCLOHEXANE	*CH
C_6H_{12}	CYCLOHEXANE	CH
C_6H_{12}	*ME-CYCLOPENTANE	*MCP
C_6H_{12}	METHYLCYCLOPENTANE	MCP
C_6H_{12}	*1-HEXENE	*1C6=
C_6H_{12}	1-HEXENE	1C6=
C_6H_{14}	2,2-DIMETHYLBUTANE	22DMB
C_6H_{14}	2,3-DIMETHYLBUTANE	23DMB
C_6H_{14}	*N-HEXANE	*NC6
C_6H_{14}	N-HEXANE	NC6
C_6H_{14}	2-METHYLPENTANE	2MC5
C_6H_{14}	3-METHYLPENTANE	3MC5
$C_6H_{14}O_4$	TRIETHYLENE GLYCOL	TRIEG
C_7H_8	*TOLUENE	*TOL
C_7H_8	TOLUENE	TOL
C_7H_8O	O-CRESOL	OCR
C_7H_{14}	*ME-CYCLOHEXANE	*MCH
C_7H_{14}	METHYLCYCLOHEXANE	MCH
C_7H_{14}	ETHYLCYCLOPENTANE	ECP
C_7H_{14}	1-HEPTENE	1C7=
C_7H_{16}	*N-HEPTANE	*NC7
C_7H_{16}	N-HEPTANE	NC7
C_8H_8	STYRENE	STYR
C_8H_{10}	*ETHYLBENZENE	*EB
C_8H_{10}	ETHYLBENZENE	EB
C_8H_{10}	XYLENES AND ETB	
C_8H_{10}	*M-XYLENE	*M-XYL
C_8H_{10}	M-XYLENE	M-XYL
C_8H_{10}	*O-XYLENE	*O-XYL
C_8H_{10}	O-XYLENE	O-XYL
C_8H_{10}	*P-XYLENE	*P-XYL
C_8H_{10}	P-XYLENE	P-XYL
C_8H_{12}	N-PROPYLBENZENE	
C_8H_{16}	ETHYLCYCLOHEXANE	ECH
C_8H_{16}	N-PROPYLCYCLOPENTA	NPCP
C_8H_{18}	*N-OCTANE	*NC8
C_8H_{18}	N-OCTANE	NC8
$C_8H_{18}O_5$	TETRAETHENE GLYCOL	TETEG
C_9H_8	INDENE	

C_9H_{10}	INDANE	
C_9H_{10}	METHYLSTYRENE	
C_9H_{12}	C3ALKYLBENZENE	C3BZ
C_9H_{12}	1-ET-2-ME-BENZENE	EMB
C_9H_{18}	N-PROPYLCYCLOHEXAN	NPCH
C_9H_{20}	*N-NONANE	*NC9
C_9H_{20}	N-NONANE	NC9
$C_{10}H_8$	NAPHTHALENE	NAPH
$C_{10}H_{10}$	1-METHYLINDENE	1M-IND
$C_{10}H_{10}$	2-METHYLINDENE	2M-IND
$C_{10}H_{12}$	DICYCLOPENTADIENE	DCPD
$C_{10}H_{14}$	N-BUTYLBENZENE	NC4BZ
$C_{10}H_{14}$	1,2-DIME-3-ETHBZ	12DMEB
$C_{10}H_{20}$	N-BUTYLCYCLOHEXANE	NBCH
$C_{10}H_{22}$	*N-DECANE	*NC10
$C_{10}H_{22}$	N-DECANE	NC10
$C_{11}H_{10}$	1-METHYLNAPHTHALEN	1MNAPH
$C_{11}H_{10}$	2-METHYLNAPHTHALEN	2MNAPH
$C_{11}H_{24}$	*N-UNDECANE	*NC11
$C_{11}H_{24}$	N-UNDECANE	NC11
$C_{12}H_8$	ACENAPHTHYLENE	
$C_{12}H_{10}$	BIPHENYL	B-P
$C_{12}H_{12}$	2,7-DIMETHYLNAPHTH	27DMN
$C_{12}H_{13}$	1,2,3-TRIME-INDENE	123TMI
$C_{12}H_{26}$	*N-DODECANE	*NC12
$C_{12}H_{26}$	N-DODECANE	NC12
$C_{13}H_{10}$	FLUORENE	
$C_{13}H_{14}$	C3ALKYLNAPHTHALENE	C3NAPH
$C_{13}H_{14}$	1-ME-4-ETH-NAPHTHA	MENAPH
$C_{13}H_{14}$	2,3,4-TRIME-NAPHTH	235TMN
$C_{13}H_{28}$	*N-TRIDECANE	*NC13
$C_{13}H_{28}$	N-TRIDECANE	NC13
$C_{14}H_{10}$	PHENANTHRENE	
$C_{14}H_{30}$	*N-TETRADECANE	*NC14
$C_{14}H_{30}$	N-TETRADECANE	NC14
$C_{15}H_{12}$	1-PHENYLINDENE	1P-IND
$C_{15}H_{14}$	2-ETHYLFLUORENE	
$C_{15}H_{32}$	*N-PENTADECANE	*NC15
$C_{15}H_{32}$	N-PENTADECANE	NC15
$C_{16}H_{10}$	FLUORANTHENE	
$C_{16}H_{10}$	PYRENE	
$C_{16}H_{12}$	1-PHENYLNAPHTHALEN	1PNAPH
$C_{16}H_{34}$	*N-HEXADECANE	*NC16
$C_{16}H_{34}$	N-HEXADECANE	NC16
$C_{18}H_{12}$	CHRYSENE	

APPENDIX B

FLOWTRAN BLOCKS
Taken from <u>FLOWTRAN Simulation -- An Introduction</u>
by Seader, <u>et.al.</u>

TYPE	BLOCK NAME	TITLE
Flash		
	IFLSH	Isothermal flash
	AFLSH	Adiabatic flash
	BFLSH	General purpose flash
	KFLSH	Isothermal three phase flash
	FLSH3	Adiabatic/isothermal three phase flash
Distillation		
	FRAKB	Rigorous distillation (KB method)
	DISTL	Shortcut distillation (Edminster)
	DSTWU	Shortcut distillation (Winn-Underwood)
	SEPR	Constant split fraction separation
	AFRAC	Rigorous distillation/absorption (matrix method)
Adsorption/Stripping		
	ABSBR	Rigorous absorber/stripper
Other Separation		
	EXTRC	Rigorous liquid-liquid extraction
Heat Exchange		
	EXCH1	Shortcut heat exchanger
	CLCN1	Shortcut cooler condenser
	DESUP	Shortcut desuperheater
	HEATR	Heat requirements
	EXCH2	Shortcut partial/total vaporizer/ condenser
	BOILR	Shortcut reboiler/intercooler
	HTR3	Threephase heater/cooler
	EXCH3	Shortcut heat exchanger
Miscellaneous Unit Operations		
	ADD	Stream addition
	MIX	Stream addition with no phase change
	SPLIT	Stream split
	PUMP	Centrifugal pump size and power
	MULPY	Stream multiplication by a parameter

	GCOMP	Compressor and turbine
	PART	Shortcut heat exchanger
Stream Convergence		
	SCVW	Bounded Wegstein stream convergence
Control		
	CNTRL	Feedback controller
	PCVB	Multiple-parameter control block
	DSPLT	Distillate feed forward control
	RCNTL	Ratio, sum and difference feedback controller
Cost Analysis		
	CAFLH	Flash drum cost
	CFLH3	
	CIFLH	
	CKFLH	
	CAFRC	Distillation
	CDSTL	
	CFRKB	
	CABSR	Packed absorber cost
	CCLN1	Heat exchanger cost
	CEXC1	
	CEXC2	
	CEXC3	
	CPUMP	Pump cost
	CCOMP	Compressor cost
	CTABS	Tray absorber cost
	CHETR	Heat exchanger cost
	BPROD	Raw material, by-product, and
	PRODT	product stream value
	RAWMT	
	PROFT	Profitability analysis
Report		
	SUMRY	Stream output editor
	TABLE	Component physical properties table
	GAMX	Liquid-activity-coefficients table
	SPRNT	Stream print block
	ASTM	Analytical distillation of a stream
	CURVE	heating and cooling curves
Reaction		
	REACT	Chemical reactor
	AREAC	Adiabatic add/subtract reactor
	XTNT	Chemical reactor (extent of reaction model)

APPENDIX C
MYINDEX, EQUAL1, EQUAL2, EQUAL4 and CORE FILES

```
C===========================================================
C
C      THE FOLLOWING ROUTINES WERE DEVELOPED BY
C      TIMOTHY E. ROCHE OF NJIT, NEWARK NJ TO PERMIT
C      THE USE OF FLOWTRAN ON FORTRAN66 COMPLIERS
C
C===========================================================
C===========================================================
C
      FUNCTION   MYINDX (C1, IPASS, C2)
C
C***********************************************************
C
C     THE INDEX FUNCTION SEARCHES FOR A SUBSTRING (C2) IN A
C     SPECIFIED CHARACTER STRING (C1), AND, IF IT FINDS THE
C     SUBSTRING, RETURNS THE SUBSTRINGS STARTING POSITION.
C     IF C2 OCCURS MORE THAN ONCE IN C1, THE STARTING
C     POSITION OF THE FIRST OCCURANCE (LEFTMOST) IS
C     RETURNED.  IF C2 DOES NOT OCCUR IN C1 THE VALUE ZERO
C     IS RETURNED.  THE SUBSTRING C2 IS LIMITED TO A LENGTH
C     OF ONE CHARACTER.
C
C       IPASS = NUMBER OF CHARACTERS IN THE STRING C1.
C
C***********************************************************
C
      INTEGER*4        C1, C2, COPY1, COPY2, MATCH1, MATCH2
      LOGICAL*1        IMAGE1, IMAGE2
      DIMENSION        C1(25), COPY1(25), IMAGE1(100),
     *                 IMAGE2(4)
      EQUIVALENCE      (COPY1(1), IMAGE1(1))
      EQUIVALENCE      (COPY2, IMAGE2(1))
C
C
      IMAX=IPASS
      IWORD=IMAX/4
      IF (IWORD*4 .LT. IMAX) IWORD=IWORD+1
C
      DO 10 I = 1, IWORD
            COPY1(I) = C1(I)
   10 CONTINUE
      COPY2 = C2
C
C
      CALL EQUAL1 (COPY2,1,1,MATCH2)
      DO 100 I = 1, IMAX
             CALL EQUAL1 (COPY1,I,I,MATCH1)
             IF (MATCH1 .EQ. MATCH2) GO TO 200
  100 CONTINUE
C
      MYINDX = 0
      GO TO 300
C
  200 MYINDX = I
C
C
  300 RETURN
      END
C==
      SUBROUTINE EQUAL1 (SOURCE,START,FINISH,TARGET)
C
C************************************************************
C
C      THE PURPOSE OF EQUAL1 IS TO TAKE DATA THAT WOULD
C      NORMALLY BE STORED IN A CHARACTER STRING USING
C      FORTRAN 77 BUT IS NOW STORED IN AN INTEGER*4 TYPE
C      VARIABLE, AND TRANSFER IT FROM A LARGE VARIABLE'S
C      SUBSTRING TO A SMALL VARIABLE.
C
C************************************************************
C
       INTEGER*4    SOURCE, TARGET, BEGIN,  ZEND,   START,
     *              FINISH, COPY1,  COPY2 , OFFSET, FUDGE,
     *              BLANK, FIXBEG,  BEGIN2
       LOGICAL*1    IMAGE1, IMAGE2
       DIMENSION    SOURCE(1), TARGET(1)
       DIMENSION    COPY1(112), COPY2(112), IMAGE1(448),
     *              IMAGE2(448)
       EQUIVALENCE (COPY1(1), IMAGE1(1)),
     *              (COPY2(1), IMAGE2(1))
C
       DATA         BLANK      /'    '/
C
C     CALCULATE THE START AND STOP POINTS WITHIN THE SOURCE
C
       FUDGE = 3
       BEGIN = START/4
       FIXBEG = BEGIN
       ZEND = FINISH/4
       IF (FINISH - ZEND*4 .NE. 0) ZEND = ZEND + 1
       IF (START - BEGIN*4 .EQ. 0) GO TO 1
       FIXBEG = BEGIN + 1
       FUDGE = -1
     1 CONTINUE
       ILEN = ZEND - FIXBEG + 1
       LLEN = FINISH - START + 1
       OFFSET = START - BEGIN*4 + FUDGE
C
C      COPY THE SOURCE
C
       DO 10 I=1, ILEN
             COPY1(I) = SOURCE(FIXBEG + I - 1)
    10 CONTINUE
C
C      BLANK OUT COPY2 FOR A CLEAN SLATE ... USING A 4
C      CHARACTER BLANK
C
       DO 25 JJ=1,112
             COPY2(JJ) = BLANK
```

```
   25 CONTINUE
C
C     TAKE SUBSTRING FROM SOURCE, PUT INTO TARGET IMAGE
C
      DO 20 J=1, LLEN
            IMAGE2(J) = IMAGE1(J + OFFSET)
   20 CONTINUE
C
C     COPY TARGET IMAGE INTO FINAL TARGET
C
      DO 30 K=1, ILEN
            TARGET(K) = COPY2(K)
   30 CONTINUE
C
C
  999 RETURN
      END
C===
      SUBROUTINE EQUAL2 (SOURCE,START,FINISH,TARGET)
C
C*********************************************************
C
C     THE PURPOSE OF EQUAL1 IS TO TAKE DATA THAT WOULD
C     NORMALLY BE STORED IN A CHARACTER STRING USING FORTRAN
C     77 BUT IS NOW STORED IN AN INTEGER*4 TYPE VARIABLE,
C     AND TRANSFER IT FROM A SMALL VARIABLE'S SUBSTRING TO A
C     LARGE VARIABLE.
C
C*********************************************************
C
      INTEGER*4   SOURCE, TARGET, BEGIN,  ZEND,   START,
     *            FINISH, COPY1,  COPY2,  OFFSET, FUDGE,
     *            BLANK,  FIXBEG
      LOGICAL*1   IMAGE1, IMAGE2
      DIMENSION   SOURCE(1),  TARGET(1)
      DIMENSION   COPY1(112), COPY2(112), IMAGE1(448),
     *            IMAGE2(448)
      EQUIVALENCE (COPY1(1), IMAGE1(1)),
     *            (COPY2(1), IMAGE2(1))
C
      DATA        BLANK      /'    '/
C
C     CALCULATE THE START AND STOP POINTS WITHIN THE SOURCE
C
      FUDGE = 3
      BEGIN = START/4
      FIXBEG = BEGIN
      ZEND = FINISH/4
      IF (FINISH - ZEND*4 .NE. 0) ZEND = ZEND + 1
      IF (START - BEGIN*4 .EQ. 0) GO TO 1
      FIXBEG = BEGIN + 1
      FUDGE = -1
    1 CONTINUE
      ILEN = ZEND - FIXBEG + 1
      LLEN = FINISH - START + 1
      OFFSET = START - BEGIN*4 + FUDGE
C
C     MAKE A COPY OF THE SOURCE VARIABLE
C
      DO 10 I=1, ILEN
           COPY1(I) = SOURCE(I)
   10 CONTINUE
C
C     MAKE A COPY OF THE TARGET VARIABLE
C
      DO 20 J=1, ILEN
            COPY2(J) = TARGET(J + FIXBEG - 1)
   20 CONTINUE
```

```
C
C     STICK SOURCE INTO TARGET SUBSTRING
C
      DO 30 K=1, LLEN
            IMAGE2(K + OFFSET) = IMAGE1(K)
   30 CONTINUE
C
C     COPY IMAGE SECTION OF TARGET BACK INTO TARGET
C
      DO 40 L=1, ILEN
            TARGET(L + FIXBEG - 1) = COPY2(L)
   40 CONTINUE
C
C
      RETURN
      END
C===
       SUBROUTINE EQUAL4 (SOURCE, START1, FINI1,
      *                   TARGET, START2, FINI2)
C
C*********************************************************
*
C      THE PURPOSE OF EQUAL4 IS TO TRANSFER CHARACTER
C      SUBSTRINGS FROM ONE VECTOR TO ANOTHER VECTOR.
C      THIS EMULATES THE EQUIVALENCE OF THE FORTRAN 77
C      CHARACTER SUBSTRINGS.  THIS IS THE DELUXE VERSION
C      THAT CHECKS FOR STRING LENGTH CONSISTANCY.  IF THE
C      STRING LENGTHS DO NOT MATCH THEN THE TARGET WILL
C      BE FILLED WITH TRAILING BLANKS OR CUT OFF AT THE
C      RIGHT MOST CHARACTER THAT WILL FIT.
C
C*********************************************************
C
       INTEGER*4 SOURCE, TARGET, BEGIN1, BEGIN2, START1,
      *          START2, ZEND1,  OFFST1, OFFST2, COPY1,
      *          COPY2,  FXBGN1, FXBGN2, FINI1,  FINI2,
      *          FUDGE1, FUDGE2, ILEN1,  ILEN2,  ZEND2
       LOGICAL*1 IMAGE1, IMAGE2, BLANK
       DIMENSION SOURCE(1), TARGET(1)
       DIMENSION COPY1(112), COPY2(112), IMAGE1(448),
      *          IMAGE2(448)
       EQUIVALENCE (COPY1(1), IMAGE1(1)),
      *            (COPY2(1), IMAGE2(1))
C
       DATA        BLANK       /'    '/
C
C
C      CALCULATE THE START AND STOP POINTS OF SOURCE AND
C      TARGET SUBSTRINGS
C
       FUDGE1 = 3
       FUDGE2 = 3
       BEGIN1 = START1/4
       FXBGN1 = BEGIN1
       BEGIN2 = START2/4
       FXBGN2 = BEGIN2
       ZEND1 = FINI1/4
       ZEND2 = FINI2/4
       IF (FINI1 - ZEND1*4 .NE. 0) ZEND1 = ZEND1 + 1
       IF (START1 - BEGIN1*4 .EQ. 0) GO TO 1
       FXBGN1 = BEGIN1 + 1
       FUDGE1 = -1
     1 CONTINUE
       IF (FINI2 - ZEND2*4 .NE. 0) ZEND2 = ZEND2 + 1
       IF (START2 - BEGIN2*4 .EQ. 0) GO TO 2
       FXBGN2 = BEGIN2 + 1
       FUDGE2 = -1
```

```
    2 CONTINUE
      ILEN1 = ZEND1 - FXBGN1 + 1
      ILEN2 = ZEND2 - FXBGN2 + 1
      LLEN1 = FINI1 - START1 + 1
      LLEN2 = FINI2 - START2 + 1
      OFFST1 = START1 - BEGIN1*4 + FUDGE1
      OFFST2 = START2 - BEGIN2*4 + FUDGE2
C
C     COPY THE TARGET INTO COPY2
C
      DO 5 L=1, ILEN2
           COPY2(L) = TARGET (L + FXBGN2 - 1)
    5 CONTINUE
C
C     COPY THE SOURCE INTO COPY1
C
      DO 10 I=1, ILEN1
            COPY1(I) = SOURCE(FXBGN1 + I - 1)
   10 CONTINUE
C
C     MOVE SUBSTRING (USING LLEN1)
C
      IF (LLEN2 .LT. LLEN1) GO TO 22
      DO 20 J=1, LLEN1
            IMAGE2(J + OFFST2) = IMAGE1(J + OFFST1)
   20 CONTINUE
      GO TO 24
   22 DO 23 JJ=1, LLEN2
            IMAGE2(JJ + OFFST2) = IMAGE1(JJ + OFFST1)
   23 CONTINUE
C
C     ADD BLANKS IF NECESSARY ... ONE (1) CHARACTER
C
   24 IF (LLEN2 .LE. LLEN1) GO TO 27
      LLEN11 = LLEN1 + 1
      DO 25 M=LLEN11, LLEN2
            IMAGE2(M + OFFST2) = BLANK
   25 CONTINUE
C
C     MAKE FINAL COPY OF TARGET
C
   27 DO 30 K=1, ILEN2
            TARGET(K + FXBGN2 - 1) = COPY2(K)
   30 CONTINUE
C
C
  999 RETURN
      END
C==
/LOGON
/REM ... SPERRY-UNIVAC 90/80-4 ... GENERATE CORE FILES
/REM
/REM ... FILE NAME "COREFILE"
/REM     THIS FILE ASSEMBLES THE MACROS THAT CREATE THE
/REM     CORE RESIDENT FILES 'DSET03' & 'DSET08'.
/REM
/REM ... CONTINUATIONS IN $ASSEMB REQUIRE A NON BLANK IN
/REM     COLUMN 72, AND THE FOLLOWING LINES STARTING IN
/REM     COLUMN 16.
/REM
/REM === /PARAM ASMLST=NO
/PARAM ASMLST=YES
/EXEC $ASSEMB
 DDS DSREF=3,RECFORM=FIXUNB,BLKSIZE=160,RECSIZE=160,
X
               TYPEFLE=INOUT,DEVICE=CORE
 DDS DSREF=8,RECFORM=FIXUNB,BLKSIZE=480,RECSIZE=480,
X
               TYPEFLE=INOUT,DEVICE=CORE
 DVLST 1,2,5,6,7,97,98,99,3,8
 END
/REM
/LOGOFF
```

Note: The non-blank character in column 72 in the DDS lines did not print because of space constraints.

APPENDIX D
PROCEDURE FILES

Sperry-Univac 90/80-4 Procedure Files

Presented are the two files, which are used to execute the program INF located in the file INF.EXE. Note that the data file name must not include the suffix '.DAT'.

```
/LOGON
/DO INF.PROC,(data-file-name)
/LOGOFF
```

```
/PROC C,(+DATAFILE)
/REM ... THE POSITIONAL PARAMETER IN THE PROCEDURE FILE IS
/REM     DATA FILE NAME WITHOUT THE '.DAT' SUFFIX
/REM          +DATAFILE ... REQUIRED PARAMETER
/REM          &DATAFILE ... OPTIONAL PARAMETER
/REM
/REM ... FILE NAME "INF.PROC"
/REM     FILE EXECUTES THE PROGRAM "INF" FOUND IN
/REM     FILE "INF.EXE"
/REM
/REM === /PROC C,(+DATAFILE)
/REM     PROC C = PRINT OUT OF PROC,
/REM     PROC N = SUPRESS PRINT OUT OF PROC
/REM
/REM     TITLE     INF PROC: EXECUTES THE PROGRAM INF FOUND
/REM                         IN FILE 'INF.EXE'
/REM
/REM     AUTHOR    A C PAULS, CED, MONSANTO CO, ST LOUIS, MO
/REM     DATE      830810
/REM
/REM     ROUTINE MODIFIED BY: T. E. ROCHE
/REM                          NJIT
/REM                          NEWARK, NJ 07102
/REM     DATE                 MARCH 1986
/REM
/REM   DATA FILE MUST BE SUBMITTED WITHOUT THE '.DAT' SUFFIX
/REM
/FSTAT &DATAFILE..DAT
/SKIP .GOTFILE
/STEP
/REM ... INVALID FILENAME ... CORRECT AND RETRY !!!!
```

```
/SKIP .ABTERM
/REM
/.GOTFILE REM ... DATA FILE HAS BEEN FOUND
/REM
/REM ... DEFINE INPUT AND OUTPUT FILES
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=&DATAFILE..DAT
/REM     SYSLST=DSET06
/SYSFILE SYSLST=&DATAFILE..INF
/REM
/REM ... CORE SCRATCH FILES NEED NOT BE DEFINED
/REM     DSET3 AND DSET8 ARE DEFINED USING AN ASSEMBLY
/REM     PROGRAM CONSISTING OF THE MACRO'S DVLST AND DDS.
/REM
/REM ... SCRATCH FILES NEED TO BE DEFINED
/REM     THE FILE COMMAND IS APPLICABLE ONLY TO DISK & TAPE
/REM     FILES CORE FILES DO NOT REQUIRE A 'FILE' COMMAND.
/FILE T.&SYSTSN..FIL.04,LINK=DSET4
/FILE T.&SYSTSN..FIL.15,LINK=DSET15
/FILE T.&SYSTSN..FIL.16,LINK=DSET16
/FILE T.&SYSTSN..FIL.17,LINK=DSET17
/REM
/REM ... FILES 18 & 19 ARE LIBRARY DATA FILES ... FILE #18
/REM     IS THE PRIVATE LIBRARY FILE, AND FILE #19 IS THE
/REM     PUBLIC LIBRARY FILE.
/REM ... OPEN=INOUT IS NEEDED SO THAT THE PRIVATE AND PUBLIC
/REM     FILES ARE NOT ALTERED BY THE FILE COMMAND.
/REM ... ACTUAL RECORD SIZE = 448 + 4 = 452 BYTES
/REM     512 BYTE RECORD SIZE USED TO MAKE DISK SPACE
/REM     SIZING EASIER.
/REM ... STANDARD BLOCK SIZE = 1024 BYTES.
/FILE FT.FTPRI.FIL,LINK=DSET18,FCBTYPE=ISAM,RECFORM=F,-
/      RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/FILE FT.FTPUB.FIL,LINK=DSET19,FCBTYPE=ISAM,RECFORM=F,-
/      RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/REM
/EXEC INF.EXE
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/ERASE T.&SYSTSN..
/REM
/PRINT &DATAFILE..INF,SPACE=E
/REM
/SKIP .ENDITAL
/STEP
/REM INF.EXE PROGRAM FAILED ... CHECK OUTPUT
/REM          ON FILE: &DATAFILE..INF
/STEP
```

```
/.ABTERM REM AN ABNORMAL TERMINATION OCCURRED
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/PRINT &DATAFILE..INF,SPACE=E
/STEP
/ERASE T.&SYSTSN..
/STEP
/.ENDITAL
/ENDPROC
```

Presented are the two files, which are used to execute the program PROPTY located in the file PROPTY.EXE. Note that the data file name must not include the suffix '.DAT'.

```
/LOGON
/DO PROPTY.PROC,(data-file-name)
/LOGOFF
```

```
/PROC C,(+DATAFILE)
/REM ... THE POSITIONAL PARAMETER IN THE PROCEDURE FILE IS
/REM     DATA FILE NAME WITHOUT THE '.DAT' SUFFIX
/REM          +DATAFILE ... REQUIRED PARAMETER
/REM          &DATAFILE ... OPTIONAL PARAMETER
/REM
/REM ... FILE NAME "PROPTY.PROC"
/REM     FILE EXECUTES THE PROGRAMS 'PROPTY' AND 'INF'.
/REM     THESE PROGRAMS ARE FOUND IN FILES 'PROPTY.EXE'
/REM     AND 'INF.EXE'.
/REM
/REM === /PROC C,(+DATAFILE)
/REM     PROC C = PRINT OUT OF PROC,
/REM     PROC N = SUPRESS PRINT OUT OF PROC
/REM
/REM     TITLE     PROPTY: EXECUTES THE PROGRAM INF FOUND
/REM                       IN FILE 'PROPTY.EXE' AND
/REM                       'INF.EXE'
/REM
/REM     AUTHOR    A C PAULS, CED,
/REM               MONSANTO CO, ST LOUIS, MO
/REM     DATE      830810
/REM
/REM     ROUTINE MODIFIED BY: T. E. ROCHE
/REM                          NJIT
/REM                          NEWARK, NJ 07102
/REM     DATE                 MARCH 1986
/REM
/REM   DATA FILE MUST BE SUBMITTED WITHOUT THE '.DAT' SUFFIX
/REM
/FSTAT &DATAFILE..DAT
/SKIP .GOTFILE
/STEP
/REM ... INVALID FILENAME ... CORRECT AND RETRY !!!!
/SKIP .ABTERM
/REM
/.GOTFILE REM ... DATA FILE HAS BEEN FOUND
/REM
```

```
/REM ... DEFINE INPUT AND OUTPUT FILES
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=&DATAFILE..DAT
/REM     SYSLST=DSET06
/SYSFILE SYSLST=&DATAFILE..PTY
/REM
/REM ... CORE SCRATCH FILES NEED NOT BE DEFINED
/REM     DSET3 AND DSET8 ARE DEFINED USING AN ASSEMBLY
/REM     PROGRAM CONSISTING OF THE MACRO'S DVLST AND DDS.
/REM
/REM ... SCRATCH FILES NEED TO BE DEFINED
/REM     THE FILE COMMAND IS APPLICABLE ONLY TO DISK & TAPE
/REM     FILES CORE FILES DO NOT REQUIRE A 'FILE' COMMAND.
/FILE T.&SYSTSN..FTII,LINK=DSET7
/REM
/REM ===
/REM ... CORRELATE PROPTY DATA (PROGRAM PROPTY)
/EXEC PROPTY.EXE
/REM
/SKIP .GOTOINF
/STEP
/SKIP .ABTERM
/REM
/.GOTOINF
/REM
/REM ... SCRATCH FILES NEED TO BE DEFINED
/REM     THE FILE COMMAND IS APPLICABLE ONLY TO DISK & TAPE
/REM     FILES. CORE FILES DO NOT REQUIRE A 'FILE' COMMAND.
/FILE T.&SYSTSN..FTPI,LINK=DSET4
/REM
/REM ... FILES 18 & 19 ARE LIBRARY DATA FILES ... FILE #18
/REM     IS THE PRIVATE LIBRARY FILE, AND FILE #19 IS THE
/REM     PUBLIC LIBRARY FILE.
/REM ... OPEN=INOUT IS NEEDED SO THAT THE PRIVATE AND PUBLIC
/REM     FILES ARE NOT ALTERED BY THE FILE COMMAND.
/REM ... ACTUAL RECORD SIZE = 448 + 4 = 452 BYTES
/REM     512 BYTE RECORD SIZE USED TO MAKE DISK SPACE
/REM     SIZING EASIER.
/REM ... STANDARD BLOCK SIZE = 1024 BYTES.
/FILE FT.FTPRI.FIL,LINK=DSET18,FCBTYPE=ISAM,RECFORM=F,-
/      RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/FILE FT.FTPUB.FIL,LINK=DSET19,FCBTYPE=ISAM,RECFORM=F,-
/      RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/REM
/REM ... REDEFINE THE INPUT DEVICE
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=T.&SYSTSN..FTII
/REM
/REM ===
/REM ... STORE PROPERTY DATA (PROGRAM INF)
```

```
/EXE INF.EXE
/REM
/REM ... APPEND FILE T.&SYSTSN..FTPI TO OUTPUT FILE
/EXEC $EDT
/TRANS START
      @READ'&DATAFILE..PTY'
      @READ'T.&SYSTSN..FTPI'
      @WRITE'&DATAFILE..PTY'
      @HALT
/TRANS END
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/ERASE T.&SYSTSN..
/REM
/PRINT &DATAFILE..VLO,SPACE=E
/REM
/SKIP .ENDITAL
/STEP
/.ABTERM REM AN ABNORMAL TERMINATION OCCURRED IN EITHER
/REM     PROPTY.EXE OR INF.EXE ... CHECK THE OUTPUT
/REM     FILE &DATAFILE..PTY
/REM
/REM ... APPEND FILE T.&SYSTSN..FTPI TO OUTPUT FILE
/EXEC $EDT
/TRANS START
      @READ'&DATAFILE..PTY'
      @READ'T.&SYSTSN..FTPI'
      @WRITE'&DATAFILE..PTY'
      @HALT
/TRANS END
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/PRINT &DATAFILE..PTY,SPACE=E
/STEP
/ERASE T.&SYSTSN..
/STEP
/.ENDITAL
/ENDPROC
```

Presented are the two files, which are used to execute the program VLE located in the file VLE.EXE. Note that the data file name must not include the suffix '.DAT'.

```
/LOGON
/DO VLE.PROC,(data-file-name)
/LOGOFF
```

```
/PROC C,(+DATAFILE)
/REM ... THE POSITIONAL PARAMETER IN THE PROCEDURE FILE IS
/REM     DATA FILE NAME WITHOUT THE '.DAT' SUFFIX
/REM          +DATAFILE ... REQUIRED PARAMETER
/REM          &DATAFILE ... OPTIONAL PARAMETER
/REM
/REM ... FILE NAME "VLE.PROC"
/REM     FILE EXECUTES THE PROGRAMS 'INF' AND 'VLE'.  THESE
/REM     PROGRAMS ARE FOUND IN FILES 'INF.EXE' AND
/REM     'VLE.EXE'.
/REM
/REM === /PROC C,(+DATAFILE)
/REM     PROC C = PRINT OUT OF PROC,
/REM     PROC N = SUPRESS PRINT OUT OF PROC
/REM
/REM     TITLE     VLE PROC: EXECUTES THE PROGRAM INF FOUND
/REM                         IN FILE 'INF.EXE' AND 'VLE.EXE'
/REM
/REM     AUTHOR    A C PAULS, CED, MONSANTO CO, ST LOUIS, MO
/REM     DATE      830810
/REM
/REM     ROUTINE MODIFIED BY: T. E. ROCHE
/REM                          NJIT
/REM                          NEWARK, NJ 07102
/REM     DATE                 MARCH 1986
/REM
/REM   DATA FILE MUST BE SUBMITTED WITHOUT THE '.DAT' SUFFIX
/REM
/FSTAT &DATAFILE..DAT
/SKIP .GOTFILE
/STEP
/REM ... INVALID FILENAME ... CORRECT AND RETRY !!!!
/SKIP .ABTERM
/REM
/.GOTFILE REM ... DATA FILE HAS BEEN FOUND
/REM
/REM ... DEFINE INPUT AND OUTPUT FILES
/REM     SYSDTA=DSET05
```

```
/SYSFILE SYSDTA=&DATAFILE..DAT
/REM     SYSLST=DSET06
/SYSFILE SYSLST=&DATAFILE..VLO
/REM
/REM ... CORE SCRATCH FILES NEED NOT BE DEFINED
/REM     DSET3 AND DSET8 ARE DEFINED USING AN ASSEMBLY
/REM     PROGRAM CONSISTING OF THE MACRO'S DVLST AND DDS.
/REM
/REM ... SCRATCH FILES NEED TO BE DEFINED
/REM     THE FILE COMMAND IS APPLICABLE ONLY TO DISK & TAPE
/REM     FILES CORE FILES DO NOT REQUIRE A 'FILE' COMMAND.
/FILE T.&SYSTSN..FTPI,LINK=DSET4
/REM
/REM ... FILES 18 & 19 ARE LIBRARY DATA FILES ... FILE #18
/REM     IS PRIVATE LIBRARY FILE, AND FILE #19 IS THE PUBLIC
/REM     LIBRARY FILE.
/REM ... OPEN=INOUT IS NEEDED SO THAT THE PRIVATE AND PUBLIC
/REM     FILES ARE NOT ALTERED BY THE FILE COMMAND.
/REM ... ACTUAL RECORD SIZE = 448 + 4 = 452 BYTES
/REM     512 BYTE RECORD SIZE USED TO MAKE DISK SPACE
/REM     SIZING EASIER.
/REM ... STANDARD BLOCK SIZE = 1024 BYTES.
/FILE FT.FTPRI.FIL,LINK=DSET18,FCBTYPE=ISAM,RECFORM=F,-
/     RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/FILE FT.FTPUB.FIL,LINK=DSET19,FCBTYPE=ISAM,RECFORM=F,-
/     RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/REM
/REM ===
/REM ... RETRIEVE PROPERTY DATA (PROGRAM INF)
/EXEC INF.EXE
/REM
/SKIP .GOTOVLE
/STEP
/SKIP .ABTERM
/REM
/.GOTOVLE
/REM
/REM ... RE-DEFINE INPUT FILE
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=T.&SYSTSN..FTPI
/REM
/REM ... CORRELATE VLE DATA (PROGRAM VLE)
/REM
/EXEC VLE.EXE
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/ERASE T.&SYSTSN..
/REM
```

```
/PRINT &DATAFILE..VLO,SPACE=E
/REM
/SKIP .ENDITAL
/STEP
/REM
/.ABTERM REM AN ABNORMAL TERMINATION OCCURRED IN EITHER
/REM      INF.EXE OR VLE.EXE ... CHECK THE OUTPUT FILE
/REM      &DATAFILE..VLO
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/PRINT &DATAFILE..VLO,SPACE=E
/STEP
/ERASE T.&SYSTSN..
/STEP
/.ENDITAL
/ENDPROC
```

Presented are the two files, which are used to execute the programs INF.EXE, PREPRO.EXE, $BGFOR (Fortran compiler), $LMR (Library Maintenance Routine), $TSOSLNK (Linkage Editor) and, FTCI.EXE. The program FTCI.EXE is the dynamically generated flowsheet simulator program FLOWTRAN. Note that the data file name must not include the suffix '.DAT'.

```
/LOGON
/DO FT.PROC,(data-file-name)
/LOGOFF
```

```
/PROC C,(+DATAFILE)
/REM ... THE POSITIONAL PARAMETER IN THE PROCEDURE FILE IS
/REM     DATA FILE NAME WITHOUT THE '.DAT' SUFFIX
/REM          +DATAFILE ... REQUIRED PARAMETER
/REM          &DATAFILE ... OPTIONAL PARAMETER
/REM
/REM ... FILE NAME "FT.PROC"
/REM     FILE EXECUTES THE PROGRAMS 'INF' AND 'PREPRO'.
/REM     THESE PROGRAMS ARE FOUND IN FILES 'INF.EXE' AND
/REM     'PREPRO.EXE'.  THE FLOWTRAN MAIN PROGRAM
/REM     (GENERATED BY PREPRO.EXE) IS COMPILED, AND
/REM     THEN LINKED WITH THE $LMR FILE (FT.OML.LIBRARY)
/REM     TO GENERATE THE LOAD MODULE (T.&SYSTSN..FTCI.EXE),
/REM     WHICH IS THE FLOWTRAN SIMULATION PROGRAM.
/REM
/REM === /PROC C,(+DATAFILE)
/REM     PROC C = PRINT OUT OF PROC,
/REM     PROC N = SUPRESS PRINT OUT OF PROC
/REM
/REM     TITLE   FT PROC: EXECUTES THE PROGRAM INF FOUND IN
/REM                      FILE 'INF.EXE' AND THE PROGRAM
/REM                      PREPRO FOUND IN FILE 'PREPRO.EXE'.
/REM                      THE GENERATED FLOWTRAN MAIN
/REM                      PROGRAM (T.&SYSTSN..FTCI.FOR) IS
/REM                      COMPILED, THEN LINKED WITH THE
/REM                      $LMR FILE TO YIELD THE FLOWTRAN
/REM                      PROGRAM T.&SYSTSN..FTCI.EXE,
/REM                      WHICH IS EXECUTED.
/REM
/REM     AUTHOR    A C PAULS, CED, MONSANTO CO, ST LOUIS, MO
/REM     DATE      830810
/REM
/REM     ROUTINE MODIFIED BY: T. E. ROCHE
/REM                          NJIT
```

```
/REM                                NEWARK, NJ 07102
/REM     DATE                       MARCH 1986
/REM
/REM   DATA FILE MUST BE SUBMITTED WITHOUT THE '.DAT' SUFFIX
/REM
/FSTAT &DATAFILE..DAT
/SKIP .GOTFILE
/STEP
/REM ... INVALID FILENAME ... CORRECT AND RETRY !!!!
/SKIP .ABTERM
/REM
/.GOTFILE REM ... DATA FILE HAS BEEN FOUND
/REM
/REM ... DEFINE INPUT AND OUTPUT FILES
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=&DATAFILE..DAT
/REM     SYSLST=DSET06
/SYSFILE SYSLST=&DATAFILE..FTO
/REM
/REM ... CORE SCRATCH FILES NEED NOT BE DEFINED
/REM     DSET3 AND DSET8 ARE DEFINED USING AN ASSEMBLY
/REM     PROGRAM CONSISTING OF THE MACRO'S DVLST AND DDS.
/REM
/REM ... SCRATCH FILES NEED TO BE DEFINED
/REM     THE FILE COMMAND IS APPLICABLE ONLY TO DISK & TAPE
/REM     FILES. CORE FILES DO NOT REQUIRE A 'FILE' COMMAND.
/FILE T.&SYSTSN..FTPI,LINK=DSET4
/REM
/REM ... FILES 18 & 19 ARE LIBRARY DATA FILES ... FILE #18
/REM     IS THE PRIVATE LIBRARY FILE, AND FILE #19 IS THE
/REM     PUBLIC LIBRARY FILE.
/REM ... OPEN=INOUT IS NEEDED SO THAT THE PRIVATE AND PUBLIC
/REM     FILES ARE NOT ALTERED BY THE FILE COMMAND.
/REM ... ACTUAL RECORD SIZE = 448 + 4 = 452 BYTES
/REM     512 BYTE RECORD SIZE USED TO MAKE DISK SPACE
/REM     SIZING EASIER.
/REM ... STANDARD BLOCK SIZE = 1024 BYTES.
/FILE FT.FTPRI.FIL,LINK=DSET18,FCBTYPE=ISAM,RECFORM=F,-
/     RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/FILE FT.FTPUB.FIL,LINK=DSET19,FCBTYPE=ISAM,RECFORM=F,-
/     RECSIZE=512,BLKSIZE=(STD,2),SPACE=(96,24),OPEN=INOUT
/REM
/REM ===
/REM ... RETRIEVE PROPERTY DATA (PROGRAM INF)
/EXEC INF.EXE
/REM
/SKIP .GOTOPRE
/STEP
/SKIP .ABTERM
/REM
```

```
/.GOTOPRE
/REM
/REM ... SCRATCH FILES NEED TO BE DEFINED
/REM      THE FILE COMMAND IS APPLICABLE ONLY TO DISK & TAPE
/REM      FILES CORE FILES DO NOT REQUIRE A 'FILE' COMMAND.
/FILE T.&SYSTSN..SCR,LINK=DSET1
/FILE FT.FTBT.FIL,LINK=DSET8
/FILE T.&SYSTSN..FTCI.FOR,LINK=DSET9
/FILE T.&SYSTSN..FTSI,LINK=DSET11
/REM
/REM ... RE-DEFINE INPUT FILE
/REM      SYSDTA=DSET05
/SYSFILE SYSDTA=T.&SYSTSN..FTPI
/REM
/REM ===
/REM ... GENERATE THE FORTRAN CODE FOR FLOWTRAN
/REM      (PROGRAM PREPRO)
/EXEC PREPRO.EXE
/REM
/SKIP .GOTOCOM
/STEP
/SKIP .ABTERM
/REM
/.GOTOCOM
/REM
/REM ... RE-DEFINE INPUT FILE
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=T.&SYSTSN..FTCI.FOR
/REM
/REM ===
/REM ... COMPILE THE FORTRAN FILE (PROGRAM $BGFOR)
/PARAM LIST=NO,MAP=NO,DEBUG=YES
/EXEC $BGFOR
/REM
/SKIP .GOTOLMR
/STEP
/SKIP .ABTERM
/REM
/.GOTOLMR
/REM
/REM ... INPUT FILE IS *
/REM
/REM ===
/REM ... PUT COMPILER OUTPUT IN FILE: T.&SYSTSN..FTCI.OBJ
/REM      (PROGRAM $LMR)
/EXEC $LMR
/TRANS START
 CONTROL OUTFILE=(T.&SYSTSN..FTCI.OBJ),-
         LISTING=(MODNAMES,SYSLST)
  COPYALL SOURCE=*
 END
/TRANS END
/REM
/SKIP .GOTOLNK
/STEP
/SKIP .ABTERM
/REM
/.GOTOLNK
/REM
/REM ... INPUT FILE ARE DEFINED VIA THE INCLUDE
/REM     STATEMENTS FOR $TSOSLNK
/REM
/REM ===
/REM ... GENERATE THE FLOWTRAN PROGRAM IN FILE:
/REM     T.&SYSTSN..FTCI.EXE (PROGRAM $TSOSLNK)
/EXEC $TSOSLNK
/TRANS START
 PROGRAM FTCI,FILENAM=T.&SYSTSN..FTCI.EXE,VERSION=66,
         ENTRY=FTCI,MAP=N
  INCLUDE FTCI,T.&SYSTSN..FTCI.OML
  INCLUDE $BLOCK,T.&SYSTSN..FTCI.OML
  INCLUDE INPUT,FT.LOAD.LIBRARY
  INCLUDE INPUTD,FT.LOAD.LIBRARY
  RESOLVE ,FT.LOAD.LIBRARY
 BIND
 END
/TRANS END
/REM
/SKIP .GOTORUN
/STEP
/SKIP .ABTERM
/REM
/.GOTORUN
/REM
/REM ... SCRATCH FILES NEED TO BE DEFINED
/REM     THE FILE COMMAND IS APPLICABLE ONLY TO DISK & TAPE
/REM     FILES. CORE FILES DO NOT REQUIRE A 'FILE' COMMAND.
/FILE T.&SYSTSN..HISTORY,LINK=DSET1
/REM
/REM ... RE-DEFINE INPUT FILE
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=T.&SYSTSN..FTSI
/REM
/REM ===
/REM ... EXECUTE THE FLOWTRAN SIMULATION
/REM     (PROGRAM T.&SYSTSN..FTCI.EXE)
/EXEC T.&SYSTSN..FTCI.EXE
/REM
/REM ... APPEND FILE T.&SYSTSN..HISTORY TO OUTPUT FILE
/EXEC $EDT
```

```
/TRANS START
      @READ'&DATAFILE..FTO'
      @READ'T.&SYSTSN..HISTORY'
      @WRITE'&DATAFILE..FTO'
      @HALT
/TRANS END
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/ERASE T.&SYSTSN..
/REM
/PRINT &DATAFILE..FTO,SPACE=E
/REM
/SKIP .ENDITAL
/STEP
/.ABTERM REM AN ABNORMAL TERMINATION OCCURRED IN EITHER
/REM     ... CHECK THE OUTPUT FILE: &DATAFILE..FTO
/REM     &DATAFILE..VLO
/REM
/REM ... APPEND FILE T.&SYSTSN..HISTORY TO OUTPUT FILE
/EXEC $EDT
/TRANS START
      @READ'&DATAFILE..FTO'
      @READ'T.&SYSTSN..HISTORY'
      @WRITE'&DATAFILE..FTO'
      @HALT
/TRANS END
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/PRINT &DATAFILE..FTO,SPACE=E
/STEP
/ERASE T.&SYSTSN..
/STEP
/.ENDITAL
/ENDPROC
```

Presented are the two files, which are used to generate the Private Data Library. The first file is the Job Control Language to create the data library, and the second is the required Fortran Main Program that creates the Private Data Library with the password "MONSANTO".

```
/LOGON
/REM
/REM ... FILE NAME IS "FT.FTPRI.LOAD"
/REM     THIS FILE GENERATES THE EMPTY PRIVATE DATA FILE
/REM     WITH THE NAME 'MONSANTO' IN THE BEGINNING OF
/REM     RECORD NUMBER 400
/REM
/REM ... TITLE     FT.FTPRI.LOAD: INITIALIZE FLOWTRAN BLOCK
/REM                              TABLE FILE
/REM
/REM     AUTHOR    A C PAULS, CED, MONSANTO CO, ST LOUIS, MO
/REM     DATE      830810
/REM
/REM ... ROUTINE MODIFIED BY: T. E. ROCHE
/REM                          NJIT
/REM                          NEWARK, NJ 07102
/REM     DATE                 MARCH 1986
/REM
/ERASE *
/STEP
/ERASE FT.FTPRI.FIL
/STEP
/ERASE FT.FTPRI.OUTPUT
/STEP
/REM
/REM ... DEFINE INPUT AND OUTPUT FILES
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=FT.FTPRI.FOR
/REM     SYSLST=DSET06
/SYSFILE SYSLST=(PRIMARY)
/REM
/PARAM LIST=NO,DEBUG=YES,MAP=NO
/EXEC $BGFOR
/REM
/REM ... RE-DEFINE INPUR FILE
/REM     SYSDTA=DSET05
/SYSFILE SYSDTA=(PRIMARY)
/REM
/REM ... OPEN=OUTPUT IS NEEDED WHEN THE FILE IS GENERATED
/REM     FILE IS ONLY WRITTEN TO, HENCE IT IS INITIALIZED
/REM     TO A SET OF NULL RECORDS
/REM
```

```
/REM ... THE RECORDSIZE HAS BEEN INCREASED TO 512 BYTES TO
/REM     ACCOUNT FOR THE LEADING "GREEN WORD" IN EACH RECORD
/REM
/FILE FT.FTPRI.FIL,LINK=DSET18,FCBTYPE=ISAM,RECFORM=F,-
/                    RECSIZE=512,BLKSIZE=(STD,2),-
/                    SPACE=(96,24),OPEN=OUTPUT
/REM
/SYSFILE SYSLST=FT.FTPRI.OUTPUT
/REM ... THE DYNAMIC LINKING LOADER (DLL) IS USED TO
/REM     EXECUTE THE COMPILED PROGRAN FTPRI.
/EXEC *
/REM
/SYSFILE SYSDTA=(PRIMARY)
/SYSFILE SYSLST=(PRIMARY)
/REM
/ERASE *
/STEP
/LOGOFF
```

```
      PROGRAM FTPRI
C     (FLOWTRAN PRIVATE FILE INITIALIZATION)
C
C     THE DATA TO BE WRITTEN TO FILE 18 IS CONTAINED IN
C     THE DATA STATEMENT "PRITAB" ... WHICH CONTAINS THE
C     NAME  ( MONSANTO )  AND A STRING OF BLANKS.  THE
C     ENTIRE FILE CONTAINS 429 RECORDS, OF WHICH THE FIRST
C     400 ARE VOID.  THE PRIVATE TABLE LIST IS STORED IN
C     RECORDS 401-429.
C
C     EACH RECORD ON THE FILE CONTAINS 448 BYTES OR 112
C     WORDS
C
C     BLANK = 448 BYTES OF BLANKS
C
C     CHARACTER  BLANK(112)
      INTEGER*4  BLANK(112)
      DATA       BLANK       /112*'    '/
C
C     CHARACTER  PRITAB(400)*32
      INTEGER*4  PRITAB(8,400)
      DATA       PRITAB      /'MONS','ANTO', 3198*'    '/
C
C     THE UNIVAC 90/80 WRITES EACH RECORD WITH A 'GREEN
C     WORD' WHICH CONTAINS THE FORMAT OF THE RECORD.  THE
C     LENGTH OF THE GREEN WORD IS FOUR (4) BYTES, THUS ONE
C     MORE WORD IS ADDED TO THE RECORD LENGTH IN THE
C     'DEFINE FILE' STATEMENT.
```

```
C
      DEFINE FILE 18  (429, 113, U, NRECNO)
C
      NRECNO=1
C
C     THE FIRST 400 RECORDS WILL HAVE 112 WORDS
      DO 10 K = 1,400
            WRITE (18'NRECNO) (BLANK(J),J=1,112)
   10 CONTINUE
C
C     THE SECOND GROUP OF RECORDS WRITTEN CONTAIN 28+1
C     RECORDS. THE FIRST 28 RECORDS CONTAIN 112 WORDS, THE
C     FIRST TWO BEING THE PASSWORD FOR THE PRIVATE FILE.
C     THE LAST RECORD (#429) CONTAINS ONLY 64 WORDS.
C
      DO 20 K = 1,28
            WRITE (18'NRECNO) ((PRITAB(J, I+14*(K-1)),
     *                         J=1,8), I=1,14)
   20 CONTINUE
C
      WRITE (18'NRECNO) ((PRITAB(J, I+392), J=1,8), I=1,8)
C
C
      STOP
      END
```

APPENDIX E
TIME/DATE Routines

Presented are the specific computer / site dependent routines for accessing the clock in the computer. These subprograms are supplied as a separate file so that the appropriate site modifications can be made, and still interface with the Flowtran code.

```
      SUBROUTINE TDATE (IDATE)
C
C ... SUBROUTINE OBTAINES THE TIME AND DATE FOR
C     DOCUMENTATION PURPOSES.
C
C
C     SUBROUTINE TDATE  (DAT)
C     ENTRY      TIMER  (TINC)
C     ENTRY      SECOND (T)
C
C
C     DOUBLE PRECISION ARG, TVALUE
C
C     CHARACTER  DAT(2)*8
C           DAT(1) = 'HH:MM:SS'
C           DAT(2) = 'MM/DD/YY'
C
      DIMENSION  IDATE(2,2), JDATE(2,2)
C
C ... OPERATING SYSTEM ROUTINES MUST BE INCLUDED THAT
C     ACCESS THE CLOCK FOR THE TIME AND DATE.
C
      DATA JDATE   /'hr:m', 'n:sc','dd/m','m/yy'/
C
C ... DATE SHOULD RETURN "DD:MM:YY"
C     CALL DATE (ARG)
C
C ... TIME SHOULD RETURN THE SYSTEM TIME AS "HH:MM:SS"
C     CALL TIME (TVALUE)
C
C ... DUMMY VALUES ARE CURRENTLY PRINTED OUT
      IDATE(1,1) = JDATE(1,1)
      IDATE(1,2) = JDATE(1,2)
      IDATE(2,1) = JDATE(2,1)
      IDATE(2,2) = JDATE(2,2)
      RETURN
C
```

```
C ... TIMER INITIALIZES THE FLOWTRAN JOB TO THE ARBITRARY
C     TIME  OF 360.0 SECONDS.
      ENTRY  TIMER (TINC)
             TINC   = 360.0
      RETURN
C
C ... SECOND SHOULD RETURN THE SYSTEM TIME IN SECONDS, THE
C     TIME HAS BEEN ARBITRARILY SET TO 0.0 SECONDS.
      ENTRY  SECOND (T)
             T      = 0.0
      RETURN
C
C
      END
C==



      SUBROUTINE CLOCK (ITIME)
C
C ... SUBROUTINE OBTAINES THE SYSTEM TIME IN SECONDS. THE
C     VARIABLE 'ITIME' IS AN INTEGER COMPATABILE WITH
C     'FLOWTRAN'.
C
      ITIME  =  SECNDS (0.0)
C
C
      RETURN
      END
C==



      FUNCTION SECNDS (T)
C
C ... SECNDS SHOULD RETURN THE SYSTEM TIME IN SECONDS,
C          IF  T .EQ. 0.0  THEN SECNDS = CURRENT TIME
C          IF  T .NE. 0.0  THEN SECNDS = LAPSE TIME
C
C ... THE VARIABLE 'T' IS SINGLE-PRECISION.
C
C ... CURRENT TIME (TIME) = NOON (12:00PM OR 43,200 SECONDS)
      IF (T .EQ. 0.0) GO TO 10
C
C ... LAPSE TIME COMPUTATION
      TSTART = T
      TIME   = 43200.0
      TLAPSE = TIME - T
      SECNDS = TLAPSE
      GO TO 99
```

```
C
C ... CURRENT TIME COMPUTATION
10    TIME = 43200.0
      SECNDS = TIME
99    RETURN
C
C
      END
C==


      SUBROUTINE SETLIM
C
C     SUBPROGRAMS USED  =  SECOND, TIMER
C
      DOUBLE PRECISION TIME, XXX, YYY
      COMMON  /XTIME/  TIME
C
C ... TIMER INITIALIZES THE FLOWTRAN JOB TO THE ARBITRARY
C     TIME OF 360.0 SECONDS.
C
C ... SECOND SHOULD RETURN THE SYSTEM TIME IN SECONDS, THE
C     TIME HAS BEEN ARBITRARILY SET TO 0.0 SECONDS.
C
C ... THE VALUE OF 'TIME' IS STORED IN COMMON /XTIME/ AND
C     USED TO MONITOR THE CPU UTILIZATION OF 'FLOWTRAN'.
      CALL  SECOND (XXX)
      CALL  TIMER  (YYY)
            TIME   = XXX + YYY
C
C
      RETURN
      END
```

APPENDIX F
Example FLOWTRAN Jobs

For the Example #1, located in Chapter 12 (Page 159) of the "User's Manual" the following files are those used by Flowtran. Each file contains a brief discription of how it is used by the Flowtran simulation program.

(1)	EX1.DAT	...	Input Data File
(2)	EX1.FTO	...	Flowtran Output File
(3)	FTCI.FOR	...	Fortran for Main Program
(4)	FTPI	...	Passed Output Data from INF.EXE / Input to PREPRO.EXE
(5)	FTSI	...	Passed Output Data from PREPRO.EXE / Input Data to FTCI.EXE

*** Ex #1, Page 159, Chapter 12, of the User's Manual ***

$$ Input Data File: EX1.DAT

```
CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159
PROPS 4 1 2 5 1
PRINT INPUT
RETR  HYDROGEN NITROGEN CO CO2
BLOCK CP1 GCOMP S1 S2
PARAM CP1  1 750 0 1 0 .8 .85
BLOCK CCP1 CCOMP CP1
PARAM CCP1  1 1 150 1 0 .015
MOLES S1  1 42672 14268 7368 7688
TEMP  S1  100
PRESS S1  650
END CASE
END JOB
```

$$ Flowtran Output Data File: EX1.FTO

```
1
 TITLE CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159

 PROPS 4 1 2 5 1
 PRINT INPUT

 RETR  HYDROGEN NITROGEN CO CO2

 BLOCK CP1 GCOMP S1 S2

 PARAM CP1 1 750 0 1 0 .8 .85

 BLOCK CCP1 CCOMP CP1

 PARAM CCP1 1 1 150 1 0 .015

 MOLES S1 1 42672 14268 7368 7688

 TEMP  S1 100

 PRESS S1 650

 END CASE

 END JOB

1
 CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159

 PHYSICAL PROPERTY OPTIONS
    ANTOINE VAPOR PRESSURE
    REDLICH-KWONG VAPOR FUGACITY
    CORRECTED HIGH TEMP LIQ FUGACITY
    IDEAL SOLUTION ACTIVITY COEF
1
 CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159


 CP1    (GCOMP)     INLET = S1      OUTLET = S2
     OUTLET PRESSURE, PSIA   =  750.00   OUTLET TEMP, DEG F    =  126.96
     ISENTROPIC TEMP, DEG F =  122.99   ISENTROPIC HORSEPOWER=  4655.5
     INDICATED HORSEPOWER    =  5482.8   BRAKE HORSEPOWER      =  6853.5


 CCP1   (CCOMP)  COST FOR UNIT CP1
     TYPE OF COMPRESSOR             1.   PMP-CMP COST INDEX        150.0
     MAT OF CONST FACTOR          1.00   UTILITIES COST, $/HP HR   0.015
     COMPRESSOR HORSEPOWER   6853.45

     COMPRESSOR CAP COST $ 535646.09   UTILITIES COST $/HR       102.80
1
 CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159

 STREAM NAME:                  S1         S2
```

```
                              LBMOL/HR  LBMOL/HR
  1 HYDROGEN                   42672.0   42672.0
  2 NITROGEN                   14268.0   14268.0
  3 CARBON MONOXIDE            7368.00   7368.00
  4 CARBON DIOXIDE             7688.00   7688.00
 TOTAL LBMOL/HR                71996.0   71996.0
 TOTAL LB/HR                  1030493.  1030493.
 1000 BTU/HR                  48083.13  62033.89
 DEGREES F                      100.00    126.96
 PSIA                          650.000   750.000
 DENSITY, LB/FT3                1.5315    1.6802
 MOLE FRAC VAPOR                1.0000    1.0000

1
 CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159

 CP1    (GCOMP) IN       T =  100.00 F,   P =  650.00 PSIA
                OUT      T =  126.96 F,   P =  750.00 PSIA
 CCP1   (CCOMP)  COST FOR UNIT CP1      CAP=535646.$, UTL= 102.80$/HR
   **END OF HISTORY
```

```
$$  Flowtran Fortran Data File, Prepared by PREPRO.EXE
    (Compiler Input $BGFOR): FTCI.FOR

      PROGRAM FTCI
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON/STREAM/NNNNS,NNNNSX
      COMMON/STREAM/S1   (10)
      DOUBLE PRECISION S1
      COMMON/STREAM/S2   (10)
      DOUBLE PRECISION S2
      COMMON/STREAM/NONE(10)
      DOUBLE PRECISION NONE
      COMMON/PARAM/NNNNP,NNNNPX,PPPPP(    15)
      COMMON/RETEN/NNNNR,NNNNRX,RRRRR(    11)
      COMMON/CONVRG/NNNNK,KKKKK(     1)
 9999 CALL INPUT
    1 CALL GCOMP(S1   ,S2   ,
     1PPPPP(     2),RRRRR(     2))
    2 CALL CCOMP(S1   ,S2   ,
     1PPPPP(     2),RRRRR(     2),
     1PPPPP(    10),RRRRR(     9))
      CALL OUTPUT
      GO TO 9999
      END
      BLOCK DATA
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON/STREAM/ NS, NSX, S(    30)
      COMMON/PARAM/  NP, NPX, P(    15)
      COMMON/RETEN/  NR, NRX, R(    11)
      COMMON/CONVRG/ NK,  KKKKK(     1)
      DATA  S/   30*0.0/
      DATA  P/   15*0.0/
      DATA  R/   11*0.0/
      DATA  KKKKK/    1*0/
      DATA  NS, NP, NR, NK/     2,    15,    11,     1/
      END
```

```
$$  Flowtran Passed Data File, Prepared by INF.EXE
    (Input to PREPRO.EXE): FTPI.DAT

PROPS 4 1 2 5 1

PRINT INPUT

COMPONENT
$01        HYDROGEN                                        58    H2
$02         2.0160000 36.686000 59.744000 190.75800 0.3206695        0.0
$03         0.66478160E+01 0.24726470E-02-0.45576350E-05 0.31177010E-08
$04        -0.66436780E-12
$05         3.2500000 0.9600000-0.2082320-1.4743800        0.0
$06         5.6026571 418.17730 474.21400
$07         0.3206700
$01        NITROGEN                                        60    N2
$02         28.016000 139.25000 227.28800 492.90400 0.2893256 0.0799987
$03         0.69471580E+01 0.66094770E-04 0.56933950E-06 0.32268620E-10
$04        -0.96832590E-13
$05         2.8000000 2.5340000 0.0361420 0.2557765-0.0907021
$06         5.3166555 1184.7968 454.53280
$07         0.2887000
$01        CARBON MONOXIDE                                 52    CO
$02         28.010000 147.04400 239.31200 507.43700 0.2891256        0.0
$03         0.69560120E+01 0.59112400E-04 0.50758090E-06 0.76411830E-09
$04        -0.65403630E-12
$05         3.1300000 2.5800000 0.0482000 0.3413000-0.2837200
$06         5.7120894 1385.8825 462.61650
$07         0.2860800
$01        CARBON DIOXIDE                                  53    CO2
$02         44.011000 350.42600 547.56200 1070.4570 0.2728975-0.0826604
$03         0.83986048E+01 0.64757663E-02-0.35550252E-05 0.11945948E-08
$04        -0.18517015E-12
$05         5.0000000 4.7510000 0.2221405 1.5728591 0.0141371
$06         6.4702320 3521.2590 455.86900
$07         0.2720000
$09               0.0       0.0       0.0       0.0       0.0 6.3800000
$99
```

```
TITLE CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159

BLOCK CP1 GCOMP S1 S2

PARAM CP1 1 750 0 1 0 .8 .85

BLOCK CCP1 CCOMP CP1

PARAM CCP1 1 1 150 1 0 .015

MOLES S1 1 42672 14268 7368 7688

TEMP  S1 100

PRESS S1 650

END CASE

END JOB


$$  Flowtran Passed Data File, Prepared by PREPRO.EXE
    (Input to FTCI.EXE): FTSI.DAT

COMPONENT
$00
    4000001000000000000000000000000       1 2 5 1 1 1 1 1 1 1 1 1
$01
          HYDROGEN                                        58    H2

$02
           2.0160000 36.686000 59.744000 190.75800 0.3206695       0.0

$03
           0.66478160E+01 0.24726470E-02-0.45576350E-05 0.31177010E-08

$04
          -0.66436780E-12

$05
           3.2500000 0.9600000-0.2082320-1.4743800       0.0

$06
           5.6026571 418.17730 474.21400

$07
           0.3206700

$01
```

```
$04
         -0.18517015E-12

$05
          5.0000000 4.7510000 0.2221405 1.5728591 0.0141371

$06
          6.4702320 3521.2590 455.86900

$07
          0.2720000

$09
                0.0       0.0       0.0       0.0       0.0 6.3800000

$99
UNAM
        2 NONE
           9 CP1     15 CCP1
SNAM
         S1    S2    NONE
TITLE
         CENTRIFUGAL COMPRESSOR CALCULATION ... Ex #1, Page 159

FPAR
        2   5              750            0              1
0
FPAR
        6   7               .8          .85
FPAR
       10  13                1          150              1
0
FPAR
       14  14             .015
FMOL
        1  1  4          42672        14268           7368
7688
TEMP
        1                  100
PRESS
        1                  650
END CASE
END JOB


 ***   End of Files for EX #1   ***
```

For the Example #2, located in Chapter 12 (Page 163) of the "User's Manual" the following files used by Flowtran are displayed.

```
          (1) EX2.DAT   ...  Input Data File
          (2) EX2.FTO   ...  Flowtran Output File
```

*** Ex #2, Page 163, Chapter 12, of the User's Manual ***

$$ Input Data File: EX2.DAT

```
TITLE ADIABATIC FLASH CALCULATION ... Ex #2, Page 163
PROPS 3 2 2 2 2
PRINT INPUT
RETR  N-BUTANE N-PENTANE N-HEXANE
BLOCK F1 AFLSH S1 6*0 S3 S2
PARAM F1 1 102.9 0 1
BLOCK CF1 CAFLH F1
PARAM CF1 6 .01 2*0 .75
MOLES S1 1 300 400 300
TEMP  S1 300
PRESS S1 257.3
END CASE
END JOB
```

$$ Flowtran Output Data File: EX2.FTO

```
1
 TITLE ADIABATIC FLASH CALCULATION ... Ex #2, Page 163

 PROPS 3 2 2 2 2

 PRINT INPUT

 RETR  N-BUTANE N-PENTANE N-HEXANE

 BLOCK F1 AFLSH S1 6*0 S3 S2

 PARAM F1 1 102.9 0 1

 BLOCK CF1 CAFLH F1

 PARAM CF1 6 .01 2*0 .75
 MOLES S1  1 300 400 300

 TEMP  S1 300

 PRESS S1 257.3

 END CASE

 END JOB

1
 ADIABATIC FLASH CALCULATION ... Ex #2, Page 163

 PHYSICAL PROPERTY OPTIONS
    CAVETT VAPOR PRESSURE
    REDLICH-KWONG VAPOR FUGACITY
    CORRECTED LIQUID FUGACITY
    SCATCHARD-HILDEBRAND ACTIVITY COEF
1
 ADIABATIC FLASH CALCULATION ... Ex #2, Page 163


 F1    (AFLSH) T =  224.35 F,   P =  102.90 PSIA,   V/F = 0.4465
MOLS/MOL
     FEEDS   = S1
     BOTTOMS = S3   , OVERHEAD = S2
     HEAT IN =  0.0000E+00 BTU/HR
     STREAM OUTPUT
                          ---------MOLES/HR---------
                             TOTAL  LIQUID   VAPOR  LBS/HR MOLE PC K-VALUE
  1 N-BUTANE              300.000 112.433 187.567 17436.0 30.0000 2.06837
  2 N-PENTANE             400.000 223.614 176.386 28860.0 40.00000.977974
  3 N-HEXANE              300.000 217.492 82.5084 25851.6 30.00000.470349

 TOTAL MOLES/HR           1000.00 553.539 446.461
 TOTAL LBS/HR                     41410.1 30737.5 72147.6

 DEGREES F                 224.35
 PSIA                      102.90
 MOLE FRAC VAPOR           0.4465
 1000 BTU/HR              1044.33-1625.29 2669.62
 BTU/LB-F                  0.5967  0.6789  0.4859
 MOLE WT                    72.15   74.81   68.85
 ACTUAL LB/CUFT            2.4995 33.1809  1.1130
 ACTUAL GPM AND CFM               155.602 460.282
 DEGREES API                        90.57
 SP.GR. AT 60 F                   0.63718
 GPM AT 60 F                      129.831
 BPD AND MMCFD AT 60,14.7         4451.34 4.06101
 CF1   (FLSHC) DRUM COST FOR FLASH UNIT F1
     VERTICAL DRUM                          LIQUID HOLDUP TIME, M      7.5
     FRACTION OF FLOODING VEL  0.50         ALLOWABLE SHELL STRESS 14000.
     SKIRT HEIGHT, FT            5.0        CORR ALLOWANCE, IN.      0.010
     EXTRA WEIGHT ITEMS, LB   1000.         SHELL COST, $/LB          0.75
     FLASH DRUM
        LIQUID LEVEL IN DRUM =  12.49 FT   DRUM LENGTH =  19.56 FT
        DRUM DIAMETER =   3.99 FT          WALL THICKNESS =    0.32 IN

     VERTICAL DRUM COST = $   4441.73
1
 ADIABATIC FLASH CALCULATION ... Ex #2, Page 163

 STREAM NAME:                  S1         S2         S3
                           LBMOL/HR   LBMOL/HR   LBMOL/HR
  1 N-BUTANE                300.000    187.567    112.433
  2 N-PENTANE               400.000    176.386    223.614
  3 N-HEXANE                300.000    82.5083    217.492
 TOTAL LBMOL/HR             1000.00    446.461    553.539
 TOTAL LB/HR                72147.6    30737.5    41410.1
 1000 BTU/HR                1044.33    2669.62   -1625.29
 DEGREES F                   300.00     224.35     224.35
 PSIA                       257.300    102.900    102.900
 DENSITY, LB/FT3            26.6145     1.1130    33.1809
 MOLE FRAC VAPOR             0.0000     1.0000     0.0000

1
 ADIABATIC FLASH CALCULATION ... Ex #2, Page 163

   **     31-MAR-8     18:32:12
 F1    (AFLSH) T =  224.35 F,   P =  102.90 PSIA,   V/F = 0.4465
               Q =  0.000E+00 BTU/HR
 CF1   (FLSHC) COST FOR UNIT F1       CAP=  4442.$
   **END OF HISTORY


 ***   End of Files for EX #2   ***
```

For the Example #3, located in Chapter 12 (Page 167) of the "User's Manual" the following files used by Flowtran are displayed.

```
        (1) EX3.DAT    ...  Input Data File
        (2) EX3.FTO    ...  Flowtran Output File
```

*** Ex #3, Page 167, Chapter 12, of the User's Manual ***

$$ Input Data File: EX3.DAT

```
TITLE VAPORIZER  CALCULATION ... Ex #3, Page 167
PROPS 3 2 2 2 2
PRINT INPUT
RETR  PROPANE N-BUTANE N-PENTANE
BLOCK H1 HEATR S1 S2
PARAM H1 1 240 2 0 0 1
BLOCK C1 CNTRL S2 H1 1
PARAM C1 1 8 .45 300 150  0 -.001 0 -20
BLOCK CV1 CURVE S1 S2
PARAM CV1 1 3*200
BLOCK CH1 CHETR H1
PARAM CH1 1 -3 3*0 150 50 150 338
BLOCK H2 SPRNT S2 6*0
MOLES S1 1 250 400 350
TEMP  S1 150
PRESS S1 202
END CASE
END JOB
```

$$ Flowtran Output Data File: EX3.FTO

```
1
 TITLE VAPORIZER  CALCULATION ... Ex #3, Page 167

 PROPS 3 2 2 2 2

 PRINT INPUT

 RETR  PROPANE N-BUTANE N-PENTANE

 BLOCK H1 HEATR S1 S2

 PARAM H1 1 240 2 0 0 1
 BLOCK C1 CNTRL S2 H1 1

 PARAM C1 1 8 .45 300 150  0 -.001 0 -20

 BLOCK CV1 CURVE S1 S2

 PARAM CV1 1 3*200

 BLOCK CH1 CHETR H1

 PARAM CH1 1 -3 3*0 150 50 150 338

 BLOCK H2 SPRNT S2 6*0

 MOLES S1 1 250 400 350

 TEMP  S1 150

 PRESS S1 202

 END CASE

 END JOB

1
 VAPORIZER  CALCULATION ... Ex #3, Page 167

 PHYSICAL PROPERTY OPTIONS
    CAVETT VAPOR PRESSURE
    REDLICH-KWONG VAPOR FUGACITY
    CORRECTED LIQUID FUGACITY
    SCATCHARD-HILDEBRAND ACTIVITY COEF
1
 VAPORIZER  CALCULATION ... Ex #3, Page 167


 H1      - HEATR - INLET = S1    , OUTLET = S2
     OUTLET TEMP =  209.02 DEG F , PRESSURE DROP =    2.00 PSI
     DUTY =  0.5271E+07 BTU/HR



 C1      FEEDBACK CONTROLLER SET MANIPULATED PARAMETER TO  2.09017E+02


 CV1     - HEATING/COOLING CURVES
     INLET STREAM  - S1        OUTLET STREAM  - S2
     BUBBLE POINT AT  200.00 PSIA  =  189.00 DEG F
     DEW POINT    AT  200.00 PSIA  =  227.05 DEG F
```

```
   TEMPERATURE  PRESSURE  STREAM ENTHALPY  TOTAL  HEAT  FRACTION
                            AT T AND P     TRANSFERRED   VAPOR
      DEG F       PSIA        BTU/HR         BTU/HR
      150.00     200.00     -5.0231E+06     0.0000E+00   0.0000
      159.75     200.00     -4.6597E+06     3.6344E+05   0.0000
      169.50     200.00     -4.2982E+06     7.2489E+05   0.0000
      179.25     200.00     -3.9310E+06     1.0922E+06   0.0000
      189.00     200.00     -3.5575E+06     1.4656E+06   0.0000
      192.33     200.00     -2.9577E+06     2.0654E+06   0.0722
      195.67     200.00     -2.3430E+06     2.6801E+06   0.1457
      199.01     200.00     -1.7115E+06     3.3116E+06   0.2207
      202.34     200.00     -1.0634E+06     3.9597E+06   0.2972
      205.68     200.00     -4.1724E+05     4.6059E+06   0.3726
      209.02     200.00      2.4835E+05     5.2715E+06   0.4500


 CH1    (EXCHC)  COST FOR EXCHANGER UNIT H1
     TYPE OF EXCHANGER     -3.00      MAT OF CONST FACTOR        1.00
     PRESSURE FACTOR        1.00      TUBE LENGTH FACTOR         1.00
     COST INDEX            150.0      UTILITIES COST FACTOR    50.000
     EXCHANGER AREA FT2    224.35      EXCHANGER DUTY BTU/HR   5271472.

     EXCHANGER COST,    $ =       3370.00
     UTILITIES COST, $/HR =          2.64


 STREAM NAME: S2
                         ---------MOLES/HR---------
                            TOTAL   LIQUID    VAPOR   LBS/HR  MOLE PC K-VALUE
  1 PROPANE               250.000 88.8767 161.123 11023.5 25.0000 2.21575
  2 N-BUTANE              400.000 214.746 185.254 23248.0 40.0000 1.05436
  3 N-PENTANE             350.000 246.377 103.623 25252.5 35.00000.514052

 TOTAL MOLES/HR           1000.00 550.000 450.000
 TOTAL LBS/HR                     34176.1 25347.9 59524.0

 DEGREES F                 209.02
 PSIA                      200.00
 MOLE FRAC VAPOR           0.4500
 1000 BTU/HR              248.352-1611.15 1859.50
 BTU/LB-F                  0.5941  0.6800  0.4783
 MOLE WT                    59.52   62.14   56.33
 ACTUAL LB/CUFT            4.2074 27.4919  1.9643
 ACTUAL GPM AND CFM               154.994 215.071
 DEGREES API                       105.55
 SP.GR. AT 60 F                   0.59692
 GPM AT 60 F                      114.378
 BPD AND MMCFD AT 60,14.7         3921.54 4.09320
1
```

```
 VAPORIZER  CALCULATION ... Ex #3, Page 167

 STREAM NAME:                  S1         S2
                            LBMOL/HR  LBMOL/HR
  1 PROPANE                  250.000   250.000
  2 N-BUTANE                 400.000   400.000
  3 N-PENTANE                350.000   350.000
 TOTAL LBMOL/HR              1000.00   1000.00
 TOTAL LB/HR                 59524.0   59524.0
 1000 BTU/HR                -5023.12    248.35
 DEGREES F                    150.00    209.02
 PSIA                        202.000   200.000
 DENSITY, LB/FT3             32.2600    0.0000
 MOLE FRAC VAPOR              0.0000    0.4500

1
 VAPORIZER  CALCULATION ... Ex #3, Page 167

   **    14-MAR-8     17:26:43
 H1      - TEMP=  240.0F, DELP=   -2.0PSI, Q= 0.103E+08BTU/HR
    *C1     -ITER  1, MANIP PARAM= 2.4000E+02, CALC VAL= 1.0000E+00
              DESIRED VAL= 4.5000E-01, NEW MANIP PARAM= 2.2500E+02
 H1      - TEMP=  225.0F, DELP=   -2.0PSI, Q= 0.931E+07BTU/HR
    *C1     -ITER  2, MANIP PARAM= 2.2500E+02, CALC VAL= 9.2973E-01
              DESIRED VAL= 4.5000E-01, NEW MANIP PARAM= 1.9500E+02
 H1      - TEMP=  195.0F, DELP=   -2.0PSI, Q= 0.256E+07BTU/HR
    *C1     -ITER  3, MANIP PARAM= 1.9500E+02, CALC VAL= 1.3084E-01
              DESIRED VAL= 4.5000E-01, NEW MANIP PARAM= 1.9917E+02
 H1      - TEMP=  199.2F, DELP=   -2.0PSI, Q= 0.334E+07BTU/HR
    *C1     -ITER  4, MANIP PARAM= 1.9917E+02, CALC VAL= 2.2438E-01
              DESIRED VAL= 4.5000E-01, NEW MANIP PARAM= 2.0834E+02
 H1      - TEMP=  208.3F, DELP=   -2.0PSI, Q= 0.512E+07BTU/HR
    *C1     -ITER  5, MANIP PARAM= 2.0834E+02, CALC VAL= 4.3157E-01
              DESIRED VAL= 4.5000E-01, NEW MANIP PARAM= 2.0906E+02
 H1      - TEMP=  209.1F, DELP=   -2.0PSI, Q= 0.528E+07BTU/HR
    *C1     -ITER  6, MANIP PARAM= 2.0906E+02, CALC VAL= 4.5129E-01
              DESIRED VAL= 4.5000E-01, NEW MANIP PARAM= 2.0917E+02
 H1      - TEMP=  209.2F, DELP=   -2.0PSI, Q= 0.531E+07BTU/HR
    *C1     -ITER  7, MANIP PARAM= 2.0917E+02, CALC VAL= 4.5421E-01
              DESIRED VAL= 4.5000E-01, NEW MANIP PARAM= 2.0902E+02
 H1      - TEMP=  209.0F, DELP=   -2.0PSI, Q= 0.527E+07BTU/HR
   **C1     -CONVERGED IN ITER  8, MANIP PARAM= 2.0902E+02
 CH1    (EXCHC) COST FOR UNIT H1        CAP=  3370.$, UTL=   2.64$/HR
   **END OF HISTORY


***   End of Files for EX #3    ***
```

For the Example #4, located in Chapter 12 (Page 167) of the "User's Manual" the following files used by Flowtran are displayed.

```
        (1) EX4.DAT    ...   Input Data File
        (2) EX4.FTO    ...   Flowtran Output File
```

*** Ex #4, Page 173, Chapter 12, of the User's Manual   ***

$$ Input Data File: EX3.DAT

TITLE VAPORIZER CALCULATION ... Ex #4, Page 173

For the Example #4, located in Chapter 12 (Page 173) of the "User's Manual" the following files used by Flowtran are displayed.

```
        (1) EX4.DAT    ...   Input Data File
        (2) EX4.FTO    ...   Flowtran Output File
```

*** Ex #4, Page 173, Chapter 12, of the User's Manual   ***

$$ Input Data File: EX4.DAT

```
TITLE DISTILLATION COLUMN CALCULATION ... Ex #4, Page 173
PROPS 5 2 2 3 2
PRINT INPUT
RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE
BLOCK D1 DISTL S1 S3 S2
PARAM D1 1 6.827 14 8 248 252 .226 .164 0 1
BLOCK C1 CNTRL S3 D1 1
PARAM C1 1 2 8 10 3 0 -.1 -.1 -20
BLOCK CD1 CDSTL D1
PARAM CD1 1 2 8*0 150 7*0 880.6 338 2*0 .01 2*0 .75 3 150 5*0 150
MOLES S1 1 30 200 370 350 50
TEMP  S1 225
PRESS S1 250
END CASE
END JOB
```

```
$$  Flowtran Output Data File: EX4.FTO

1
 TITLE DISTILLATION COLUMN CALCULATION ... Ex #4, Page 173

 PROPS 5 2 2 3 2

 PRINT INPUT

 RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE

 BLOCK D1 DISTL S1 S3 S2

 PARAM D1 1 6.827 14 8 248 252 .226 .164 0 1

 BLOCK C1 CNTRL S3 D1 1

 PARAM C1 1 2 8 10 3 0 -.1 -.1 -20

 BLOCK CD1 CDSTL D1

 PARAM CD1 1 2 8*0 150 7*0 880.6 338 2*0 .01 2*0 .75 3 150 5*0 150

 MOLES S1 1 30 200 370 350 50

 TEMP  S1 225

 PRESS S1 250

 END CASE

 END JOB

1
 DISTILLATION COLUMN CALCULATION ... Ex #4, Page 173

 PHYSICAL PROPERTY OPTIONS
    CAVETT VAPOR PRESSURE
    REDLICH-KWONG VAPOR FUGACITY
    CHAO-SEADER LIQUID FUGACITY
    SCATCHARD-HILDEBRAND ACTIVITY COEF
1
 DISTILLATION COLUMN CALCULATION ... Ex #4, Page 173


 D1       - DISTL - PART. CONDENSER
     FEED = S1    , BOTTOMS = S3    , OVERHEAD = S2
     REFLUX RATIO=  5.44, NO. OF PLATES  =   14., FEED PLATE=  8.
     FRAC OVHD    = 0.226, FEED FRAC VAPOR= 0.164
     CONDENSER DUTY= 0.8688E+07BTU/HR, TEMP= 118.22F, PRES= 248.00PSIA
```

```
      REBOILER  DUTY= 0.9377E+07BTU/HR, TEMP= 265.34F, PRES= 252.00PSIA


 C1     FEEDBACK CONTROLLER SET MANIPULATED PARAMETER TO  5.44473E+00


 CD1    (DISTC)  COST FOR UNIT D1
      NUMBER OF DIAMETERS          2.      REFLUX RATIO                5.445
      NUMBER OF TRAYS             14.0     TYPE OF TRAY                   3.
      MOC FACTOR FOR TRAYS        1.00     TRAY COST UPDATE FACTOR   150.0
      LOADING AT TOP              0.85     LOADING AT BOTTOM            0.85
      PLATE SPACING AT TOP         24.     PLATE SPACING AT BOTTOM      18.
      SURFACE TENSION AT TOP       20.     SURFACE TENSION AT BOTTOM   20.
      FOAMING FACTOR AT TOP       1.00     FOAMING FACTOR AT BOTTOM   1.00
      OVERALL U FOR CONDENSER    150.0     COOLANT TEMP IN             80.00
      COOLANT TEMP RISE          20.00     MINIMUM TEMP APPROACH      10.00
      COOLANT HEAT CAPACITY       1.00     COOLANT SPECIFIC GRAVITY   1.00
      CONDENSATE SUBCOOLING       0.00     PRESSURE FACTOR-CONDENSER 1.00
      TYPE OF CONDENSER            -1.     TUBE LENGTH FACTOR-COND    1.00
      MOC FACTOR FOR CONDENSER    1.00     CONDENSER UPDATE FACTOR,  150.0
      COOLANT COST-C/MGL          3.00     HEATING FLUID COST-C/MLB  56.00
      HEATING FLUID MAX TEMP    338.00     REBOILER HEAT FLUX        12000.
      MIN TEMP DIF FORFLUX        1.00     HEATING FLUID DELTA H        881.
      TYPE OF REBOILER             -1.     MOC FACTOR FOR REBOILER    1.00
      PRESSURE FACTOR-REBOILER    1.00     TUBE LENGTH FACTOR-REBL    1.00
      REBOILER UPDATE FACTOR     124.0     SKIRT HEIGHT                10.0
      STRESS IN COLUMN SHELL    14000.     CORROSION ALLOWANCE        0.010
      EXTRA WEIGHT ITEMS         2500.     SHELL COST,$/LB             0.75

      CONDENSER-
         CONDENSER DUTY, BTU/HR =     0.86882041E+07
         LOG MEAN TEMP DIFFERENCE =    26.99 DEG F
         AREA = 2145.70 SQ FT
         COOLANT FLOW RATE =   868.820 GPM
         CONDENSER COST = $    13570.00
         COOLANT COST = $/HR          1.56

      REBOILER -
         REBOILER DUTY, BTU/HR =     0.93765808E+07
         AREA =  781.38 SQ FT
         STEAM FLOW RATE = 10647.945 LB/HR
         REBOILER COST = $     5440.00
         STEAM COST = $/HR          5.96

      COLUMN DIAMETERS-        COLUMN HEIGHT, FT     =  34.00
         BELOW FEED  5.5000    SHELL THICKNESS, IN  = 0.8125
         ABOVE FEED  4.0000    SHELL THICKNESS, IN  = 0.5625
         COST OF COLUMN SHELL = $    15365.00
         TRAY COST = $     2240.00
      TOTAL UTILITY COST= $/HR         7.53

      COLUMN CAPITAL COST   = $    36615.00
1
 DISTILLATION COLUMN CALCULATION ... Ex #4, Page 173

 STREAM NAME:                  S1        S2        S3
                            LBMOL/HR  LBMOL/HR  LBMOL/HR
  1 *ETHANE                 30.0000   29.9999   0.00015
  2 *PROPANE                200.000   192.039   7.96116
  3 *N-BUTANE               370.000   3.95601   366.044
  4 *N-PENTANE              350.000   0.00530   349.995
  5 *N-HEXANE               50.0000   0.00000   50.0000
 TOTAL LBMOL/HR             1000.00   226.000   774.000
 TOTAL LB/HR                60786.3   9600.10   51186.2
 1000 BTU/HR                -772.83    269.20   -353.65
 DEGREES F                   225.00    118.22    265.34
 PSIA                       250.000   248.000   252.000
 DENSITY, LB/FT3             0.0000    2.2118   27.9168
 MOLE FRAC VAPOR             0.2295    1.0000    0.0000

1
 DISTILLATION COLUMN CALCULATION ... Ex #4, Page 173

   **    14-MAR-8     17:28:28
 D1      -OVD T= 116.9F,FD T= 197.4F,BTM T= 265.7F
    *C1     -ITER  1, MANIP PARAM= 6.8270E+00, CALC VAL= 6.3366E+00
              DESIRED VAL= 8.0000E+00, NEW MANIP PARAM= 6.1270E+00
 D1      -OVD T= 117.5F,FD T= 201.1F,BTM T= 265.6F
    *C1     -ITER  2, MANIP PARAM= 6.1270E+00, CALC VAL= 6.9957E+00
              DESIRED VAL= 8.0000E+00, NEW MANIP PARAM= 5.0605E+00
 D1      -OVD T= 118.8F,FD T= 207.5F,BTM T= 265.2F
    *C1     -ITER  3, MANIP PARAM= 5.0605E+00, CALC VAL= 8.7228E+00
              DESIRED VAL= 8.0000E+00, NEW MANIP PARAM= 5.4447E+00
 D1      -OVD T= 118.2F,FD T= 205.1F,BTM T= 265.3F
   **C1     -CONVERGED IN ITER  4, MANIP PARAM= 5.4447E+00
 CD1    (DISTC)  COST FOR UNIT D1       CAP= 36615.$, UTL=   7.53$/HR
   **END OF HISTORY


 ***   End of Files for EX #4   ***
```

For the Example #4 (Modified), located in Chapter 12 (Page 173) of the "User's Manual" the following files used by Flowtran are displayed.

```
        (1) EX4A.DAT    ...  Input Data File
        (2) EX4A.FTO    ...  Flowtran Output File
```

*** Ex #4A, Page 173, Chapter 12, of the User's Manual ***

```
$$  Input Data File: EX4A.DAT

TITLE DISTILLATION COLUMN CALCULATION ... WINN UNDERWOOD METHOD
PROPS 5 2 2 3 2
PRINT INPUT
RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE
BLOCK D1 DSTWU S1 S3 S2
PARAM D1 1 3 2 82.5 27.0 .164 6.827 14 248 252 1
BLOCK C1 CNTRL S3 D1 6
PARAM C1 1 2 8 20 3 0 -.1 -.1 -20
MOLES S1 1 30 200 370 350 50
TEMP  S1 225
PRESS S1 250
END CASE
END JOB
```

```
  Flowtran Output Data File: EX4A2.FTO

1
 TITLE DISTILLATION COLUMN CALCULATION ... WINN UNDERWOOD METHOD

 PROPS 5 2 2 3 2

 PRINT INPUT

 RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE

 BLOCK D1 DSTWU S1 S3 S2

 PARAM D1 1 3 2 82.5 27.0 .164 6.827 14 248 252 1

 BLOCK C1 CNTRL S3 D1 6

 PARAM C1 1 2 8 20 3 0 -.1 -.1 -20
 MOLES S1 1 30 200 370 350 50

 TEMP  S1 225

 PRESS S1 250

 END CASE

 END JOB

1
 DISTILLATION COLUMN CALCULATION ... WINN UNDERWOOD METHOD

 PHYSICAL PROPERTY OPTIONS
    CAVETT VAPOR PRESSURE
    REDLICH-KWONG VAPOR FUGACITY
    CHAO-SEADER LIQUID FUGACITY
    SCATCHARD-HILDEBRAND ACTIVITY COEF
1
 DISTILLATION COLUMN CALCULATION ... WINN UNDERWOOD METHOD

 D1     (DSTWU)  FEED=S1   OVD=S2   BOT=S3
     HVY KEY COMP. NO.     3.000 LT. KEY COMP. NO.      2.000
     SPLIT FOR LT. KEY    82.500 SPLIT FOR HVY KEY     27.000
     QUALITY OF FEED       0.164
     DESIRED REFLUX        6.827 DESIRED NO OF STGS    14.000
     TOP PRESS. PSIA     248.000 BOTTOM PRESS. PSIA   252.000
     CONDENSER TYPE        1.000

     MIN. NO. THEO. STAGES AT TOTAL REFLUX        9.97
     MINIMUM REFLUX AT INFINITE STAGES            4.95
     NO. STAGES ABOVE FEED AT TOTAL REFLUX        5.78
     ACTUAL REFLUX                                8.78
     NO. OF THEO. STAGES AT ACTUAL REFLUX        14.00
     CONDENSER TEMP, DEG F                      118.21
     REBOILER TEMP, DEG F                       265.34


 C1      FEEDBACK CONTROLLER SET MANIPULATED PARAMETER TO  6.82700E+00
1
 DISTILLATION COLUMN CALCULATION ... WINN UNDERWOOD METHOD

 STREAM NAME:                  S1         S2         S3
                           LBMOL/HR   LBMOL/HR   LBMOL/HR
  1 *ETHANE                 30.0000    29.9998    0.00017
  2 *PROPANE                200.000    192.027    7.97269
  3 *N-BUTANE               370.000    3.95827    366.042
  4 *N-PENTANE              350.000    0.00351    349.996
  5 *N-HEXANE               50.0000    0.00000    50.0000
 TOTAL LBMOL/HR             1000.00   225.989   774.011
 TOTAL LB/HR                60786.3   9599.59   51186.7
 1000 BTU/HR                -772.83    269.17   -353.77
 DEGREES F                   225.00    118.21    265.34
 PSIA                       250.000   248.000   252.000
 DENSITY, LB/FT3             0.0000    2.2118   27.9168
 MOLE FRAC VAPOR             0.2295    1.0000    0.0000

1
 DISTILLATION COLUMN CALCULATION ... WINN UNDERWOOD METHOD

   **    21-FEB-8     13:28:02
 D1    (DSTWU)  FEED=S1   OVD=S2   BOT=S3
              ITR TOP TEMP BOT TEMP MIN THEO STGS
                1  224.517  254.517       10.087
                2  118.219  265.336        9.966
              MINIMUM REFLUX    4.947
   **C1    -CONVERGED IN ITER  1, MANIP PARAM= 6.8270E+00
   **END OF HISTORY


 ***   End of Files for EX #4A   ***
```

For the Example #4B (Modified), located in Chapter 12 (Page 173) of the "User's Manual" the following files used by Flowtran are displayed.

```
(1) EX4B.DAT   ...  Input Data File
(2) EX4B.FTO   ...  Flowtran Output File
```

*** Ex #4B, Page 173, Chapter 12, of the User's Manual ***

$$ Input Data File: EX4B.DAT

```
TITLE DISTILLATION COLUMN CALCULATION ... KB METHOD
PROPS 5 2 2 3 2
PRINT INPUT
RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE
BLOCK D1 FRAKB S1 0 0 S3 0 0 0 S2
PARAM D1 1 16 6.827 .226 1 0 248 249 252  0 118 265 1 1 9
BLOCK C1 CNTRL S3 D1 2
PARAM C1 1 2 8 10 3 0 -.1 -20
BLOCK CD1 CFRKB D1
PARAM CD1 1 2 8*0 150 7*0 880.6 338 2*0 .01 2*0 .75 3 336.3 5*0
            336.3 4*0 336.3 0 300
MOLES S1 1 30 200 370 350 50
TEMP  S1 225
PRESS S1 250
END CASE
END JOB
```

$$ Flowtran Output Data File: EX4B.FTO

```
1
 TITLE DISTILLATION COLUMN CALCULATION ... KB METHOD

 PROPS 5 2 2 3 2

 PRINT INPUT

 RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE

 BLOCK D1 FRAKB S1 0 0 S3 0 0 0 S2

 PARAM D1 1 16 6.827 .226 1 0 248 249 252  0 118 265 1 1 9
 BLOCK C1 CNTRL S3 D1 2

 PARAM C1  1 2 8 10 3 0 -.1 -20

 BLOCK CD1 CFRKB D1

 PARAM CD1  1 2 8*0 150 7*0 880.6 338 2*0 .01 2*0 .75 3 336.3 5*0

             336.3 4*0 336.3 0 300

 MOLES S1  1 30 200 370 350 50

 TEMP  S1 225

 PRESS S1 250

 END CASE

 END JOB
1
 DISTILLATION COLUMN CALCULATION ... KB METHOD

 PHYSICAL PROPERTY OPTIONS
    CAVETT VAPOR PRESSURE
    REDLICH-KWONG VAPOR FUGACITY
    CHAO-SEADER LIQUID FUGACITY
    SCATCHARD-HILDEBRAND ACTIVITY COEF
1
 DISTILLATION COLUMN CALCULATION ... KB METHOD


 D1      - FRAKB
     REFLUX RATIO       6.057   CONDENSER DUTY     0.80374E+07
     REFLUX RATE     1368.960   REBOILER DUTY      0.87255E+07
     NUMBER OF STAGES         16
     FEED  1 IS S1     ON STAGE   9   BOTTOMS IS S3
     VAPOR DISTILLATE IS S2      PC EFF IS 100.00 FOR ALL STAGES



 C1      FEEDBACK CONTROLLER SET MANIPULATED PARAMETER TO  6.05735E+00


 CD1    (DISTC)  COST FOR UNIT D1
     NUMBER OF DIAMETERS          2.     REFLUX RATIO              6.057
     NUMBER OF TRAYS           14.0     TYPE OF TRAY                 3.
     MOC FACTOR FOR TRAYS      1.00     TRAY COST UPDATE FACTOR  336.3
     LOADING AT TOP            0.85     LOADING AT BOTTOM         0.85
     PLATE SPACING AT TOP       24.     PLATE SPACING AT BOTTOM    18.
    SURFACE TENSION AT TOP      20.      SURFACE TENSION AT BOTTOM  20.
    FOAMING FACTOR AT TOP      1.00      FOAMING FACTOR AT BOTTOM  1.00
    OVERALL U FOR CONDENSER  150.0       COOLANT TEMP IN          80.00
    COOLANT TEMP RISE         20.00      MINIMUM TEMP APPROACH    10.00
    COOLANT HEAT CAPACITY      1.00      COOLANT SPECIFIC GRAVITY  1.00
    CONDENSATE SUBCOOLING      0.00      PRESSURE FACTOR-CONDENSER 1.00
    TYPE OF CONDENSER           -1.      TUBE LENGTH FACTOR-COND   1.00
    MOC FACTOR FOR CONDENSER   1.00      CONDENSER UPDATE FACTOR, 336.3
    COOLANT COST-C/MGL         3.00      HEATING FLUID COST-C/MLB300.00
    HEATING FLUID MAX TEMP   338.00      REBOILER HEAT FLUX      12000.
    MIN TEMP DIF FORFLUX       1.00      HEATING FLUID DELTA H     881.
    TYPE OF REBOILER            -1.      MOC FACTOR FOR REBOILER   1.00
    PRESSURE FACTOR-REBOILER   1.00      TUBE LENGTH FACTOR-REBL   1.00
    REBOILER UPDATE FACTOR    336.3      SKIRT HEIGHT              10.0
    STRESS IN COLUMN SHELL   14000.      CORROSION ALLOWANCE      0.010
    EXTRA WEIGHT ITEMS        2500.      SHELL COST,$/LB           0.75

    CONDENSER-
       CONDENSER DUTY, BTU/HR =     0.80374033E+07
       LOG MEAN TEMP DIFFERENCE =    27.06 DEG F
       AREA = 1980.43 SQ FT
       COOLANT FLOW RATE =   803.740 GPM
       CONDENSER COST = $    28740.00
       COOLANT COST = $/HR          1.45

    REBOILER -
       REBOILER DUTY, BTU/HR =     0.87254526E+07
       AREA =  727.12 SQ FT
       STEAM FLOW RATE =  9908.531 LB/HR
       REBOILER COST = $    14020.00
       STEAM COST = $/HR        29.73

    COLUMN DIAMETERS-        COLUMN HEIGHT, FT     =  34.00
       BELOW FEED  5.0000    SHELL THICKNESS, IN   = 0.7500
       ABOVE FEED  4.0000    SHELL THICKNESS, IN   = 0.5625
       COST OF COLUMN SHELL = $     13595.00
       TRAY COST = $      4450.00

     TOTAL UTILITY COST= $/HR        31.17

     COLUMN CAPITAL COST    = $    60805.00
1
 DISTILLATION COLUMN CALCULATION ... KB METHOD


 STREAM NAME:                 S1         S2         S3
                           LBMOL/HR   LBMOL/HR   LBMOL/HR
  1 *ETHANE                 30.0000    29.9998    0.00017
  2 *PROPANE                200.000    191.964    8.03589
  3 *N-BUTANE               370.000    4.03059    365.969
  4 *N-PENTANE             350.000   0.00551   349.995
  5 *N-HEXANE              50.0000   0.00000   50.0000
 TOTAL LBMOL/HR            1000.00   226.000   774.000
 TOTAL LB/HR               60786.3   9601.15   51185.2
 1000 BTU/HR               -772.83    269.50   -354.28
 DEGREES F                  225.00    118.28    265.32
 PSIA                      250.000   248.000   252.000
 DENSITY, LB/FT3            0.0000    2.2117   27.9166
 MOLE FRAC VAPOR            0.2295    1.0000    0.0000

1
 DISTILLATION COLUMN CALCULATION ... KB METHOD


 D1      - FEED  1 IS S1      ON STAGE   9
      BASE COMPONENT FOR KB METHOD IS COMPONENT  3
           MAX               MAX
    ITR MASS BAL CP STG HEAT BAL STG    PHI
         ERROR             ERROR
      1 -0.7E+00  2  13  0.0E+00   0  1.3999
      2  0.6E+00  2   9  0.0E+00   0  1.4303
      3 -0.2E+00  2   8  0.1E+07   1  1.4290
      4  0.1E+00  1   9  0.2E+06  10  1.2276
      5  0.2E-01  1  11 -0.2E+06   9  0.9739
      6  0.7E-02  1   9  0.5E+05   9  1.0133
      7  0.2E-02  1  11 -0.1E+05   9  1.0001
      8  0.6E-03  1  11  0.9E+03   9  1.0005
      9  0.2E-03  1  11 -0.4E+03   9  1.0001
     10  0.4E-04  1  11 -0.9E+02   9  1.0000
   **D1      - CONVERGED
     TIME=     0
    *C1     -ITER  1, MANIP PARAM= 6.8270E+00, CALC VAL= 6.7011E+00
            DESIRED VAL= 8.0000E+00, NEW MANIP PARAM= 6.1270E+00
 D1      - FEED  1 IS S1      ON STAGE   9
      BASE COMPONENT FOR KB METHOD IS COMPONENT  3
           MAX               MAX
    ITR MASS BAL CP STG HEAT BAL STG    PHI
         ERROR             ERROR
      1 -0.2E+00  2   9  0.0E+00   0  0.9655
      2 -0.3E-02  2  15  0.0E+00   0  0.9986
      3 -0.2E-02  1   9 -0.9E+06  15  0.9975
      4  0.1E+00  1   9  0.2E+06   9  1.1964
      5  0.2E-01  1  11 -0.2E+06   9  0.9775
      6  0.8E-02  1  11  0.4E+05   9  1.0116
      7  0.2E-02  1  11 -0.1E+05   9  0.9998
      8  0.6E-03  1  11  0.1E+04   9  1.0005
      9  0.2E-03  1  11 -0.4E+03   9  1.0001
     10  0.4E-04  1  12 -0.5E+02   9  1.0000
   **D1      - CONVERGED
     TIME=     0
   *C1     -ITER  2, MANIP PARAM= 6.1270E+00, CALC VAL= 7.8824E+00
             DESIRED VAL= 8.0000E+00, NEW MANIP PARAM= 6.0573E+00
D1      - FEED  1 IS S1      ON STAGE   9
     BASE COMPONENT FOR KB METHOD IS COMPONENT  3
          MAX                MAX
   ITR MASS BAL CP STG HEAT BAL STG   PHI
        ERROR             ERROR
     1 -0.2E+00  2   9  0.0E+00   0  0.9675
     2  0.3E-02  2  10  0.0E+00   0  1.0009
     3 -0.2E-03  2   9 -0.9E+05  15  0.9997
     4  0.1E-01  1   9  0.2E+05   9  1.0185
     5  0.3E-02  1  11 -0.2E+05   9  0.9975
     6  0.8E-03  1  11  0.5E+04   9  1.0012
     7  0.2E-03  1  11 -0.1E+04   9  1.0000
     8  0.7E-04  1  11  0.2E+03   9  1.0001
     9  0.2E-04  1  11 -0.6E+02   9  1.0000
  **D1      - CONVERGED
    TIME=     0
  **C1     -CONVERGED IN ITER  3, MANIP PARAM= 6.0573E+00
CD1    (DISTC)  COST FOR UNIT D1        CAP= 60805.$, UTL=  31.17$/HR
  **END OF HISTORY


***   End of Files for EX #4B   ***
```

For the Example #4C (Modified), located in Chapter 12 (Page 173) of the "User's Manual" the following files used by Flowtran are displayed.

```
        (1) EX4C.DAT    ...  Input Data File
        (2) EX4C.FTO    ...  Flowtran Output File
```

*** Ex #4C, Page 173, Chapter 12, of the User's Manual ***

$$ Input Data File: EX4C.DAT

```
TITLE DISTILLATION COLUMN CALCULATION ... RIGOROUS MATRIX METHOD
PROPS 5 2 2 3 2
PRINT INPUT
RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE
BLOCK D1 AFRAC S1 3*0 S3 4*0 S2
PARAM D1 1 264 118 252 248  1200 16 1 3 .226 6.827 9
BLOCK C1 CNTRL S3 D1 10
PARAM C1 1 2 8 10 3 0 -.1 -20
BLOCK CD1 CAFRC D1
PARAM CD1 1 2 8*0 150 7*0 880.6 338 2*0 .01 2*0 .75 3 336.3 5*0
             336.3 4*0 336.3 0 300
MOLES S1  1 30 200 370 350 50
TEMP  S1  225
PRESS S1  250
END CASE
END JOB
```

$$ Flowtran Output Data File: EX4C.FTO

```
1
 TITLE DISTILLATION COLUMN CALCULATION ... RIGOROUS MATRIX METHOD

 PROPS 5 2 2 3 2

 PRINT INPUT

 RETR  *ETHANE *PROPANE *N-BUTANE *N-PENTANE *N-HEXANE

 BLOCK D1 AFRAC S1 3*0 S3 4*0 S2

 PARAM D1 1 264 118 252 248  1200 16 1 3 .226 6.827 9
 BLOCK C1 CNTRL S3 D1 10

 PARAM C1 1 2 8 10 3 0 -.1 -20

 BLOCK CD1 CAFRC D1

 PARAM CD1 1 2 8*0 150 7*0 880.6 338 2*0 .01 2*0 .75 3 336.3 5*0

             336.3 4*0 336.3 0 300

 MOLES S1 1 30 200 370 350 50

 TEMP  S1 225

 PRESS S1 250

 END CASE

 END JOB

1
 DISTILLATION COLUMN CALCULATION ... RIGOROUS MATRIX METHOD

 PHYSICAL PROPERTY OPTIONS
    CAVETT VAPOR PRESSURE
    REDLICH-KWONG VAPOR FUGACITY
    CHAO-SEADER LIQUID FUGACITY
    SCATCHARD-HILDEBRAND ACTIVITY COEF
1
 DISTILLATION COLUMN CALCULATION ... RIGOROUS MATRIX METHOD

 D1      - AFRAC - RIGOROUS DISTILLATION/ABSORPTION
     FEED 1 IS S1   ON STAGE  9
     BOTTOM PRODUCT IS S3
     OVERHEAD VAPOR IS S2
     CONDENSER DUTY, MILLION BTU/HR   -8.0307
     REBOILER  DUTY, MILLION BTU/HR    8.7187
     STAGE   TEMP   PRESSURE   LIQ FLOW    VAP FLOW
              F       PSIA      LB-M/HR     LB-M/HR
       16   118.28   248.00     1368.36      226.00
       15   127.43   248.27     1349.45     1594.36
       14   136.57   248.53     1301.99     1575.45
       13   148.71   248.80     1242.71     1527.99
       12   164.08   249.07     1172.27     1468.71
       11   180.92   249.33     1112.10     1398.27
       10   198.07   249.60     1056.70     1338.10
        9   216.25   249.87     1834.10     1282.70
        8   222.87   250.13     1869.21     1060.10
        7   228.48   250.40     1898.79     1095.21
     6    233.51    250.67      1926.23      1124.79
     5    238.14    250.93      1951.01      1152.23
     4    242.63    251.20      1970.48      1177.01
     3    247.60    251.47      1980.60      1196.48
     2    254.35    251.73      1975.86      1206.60
     1    265.32    252.00       774.00      1201.86


C1      FEEDBACK CONTROLLER SET MANIPULATED PARAMETER TO  6.05469E+00


CD1    (CAFRC)  COST FOR UNIT D1
    NUMBER OF DIAMETERS           2.     REFLUX RATIO                 6.055
    NUMBER OF TRAYS              14.0    TYPE OF TRAY                    3.
    MOC FACTOR FOR TRAYS         1.00    TRAY COST UPDATE FACTOR      336.3
    LOADING AT TOP               0.85    LOADING AT BOTTOM             0.85
    PLATE SPACING AT TOP          24.    PLATE SPACING AT BOTTOM        18.
    SURFACE TENSION AT TOP        20.    SURFACE TENSION AT BOTTOM      20.
    FOAMING FACTOR AT TOP        1.00    FOAMING FACTOR AT BOTTOM      1.00
    OVERALL U FOR CONDENSER     150.0    COOLANT TEMP IN              80.00
    COOLANT TEMP RISE           20.00    MINIMUM TEMP APPROACH        10.00
    COOLANT HEAT CAPACITY        1.00    COOLANT SPECIFIC GRAVITY      1.00
    CONDENSATE SUBCOOLING        0.00    PRESSURE FACTOR-CONDENSER     1.00
    TYPE OF CONDENSER             -1.    TUBE LENGTH FACTOR-COND       1.00
    MOC FACTOR FOR CONDENSER     1.00    CONDENSER UPDATE FACTOR,     336.3
    COOLANT COST-C/MGL           3.00    HEATING FLUID COST-C/MLB300.00
    HEATING FLUID MAX TEMP     338.00    REBOILER HEAT FLUX          12000.
    MIN TEMP DIF FORFLUX         1.00    HEATING FLUID DELTA H         881.
    TYPE OF REBOILER              -1.    MOC FACTOR FOR REBOILER       1.00
    PRESSURE FACTOR-REBOILER     1.00    TUBE LENGTH FACTOR-REBL       1.00
    REBOILER UPDATE FACTOR      336.3    SKIRT HEIGHT                  10.0
    STRESS IN COLUMN SHELL     14000.    CORROSION ALLOWANCE          0.010
    EXTRA WEIGHT ITEMS          2500.    SHELL COST,$/LB               0.75

    CONDENSER-
       CONDENSER DUTY, BTU/HR =    0.80307010E+07
       LOG MEAN TEMP DIFFERENCE =   36.52 DEG F
       AREA = 1466.04 SQ FT
       COOLANT FLOW RATE =   803.070 GPM
       CONDENSER COST = $    23170.00
       COOLANT COST = $/HR         1.45

    REBOILER -
       REBOILER DUTY, BTU/HR =    0.87187435E+07
       AREA =  726.56 SQ FT
       STEAM FLOW RATE =  9900.912 LB/HR
       REBOILER COST = $    14010.00
       STEAM COST = $/HR       29.70
     COLUMN DIAMETERS-       COLUMN HEIGHT, FT     =  34.00
        BELOW FEED  5.0000    SHELL THICKNESS, IN   = 0.7500
        ABOVE FEED  4.0000    SHELL THICKNESS, IN   = 0.5625
        COST OF COLUMN SHELL = $     13595.00
        TRAY COST = $      4450.00

      TOTAL UTILITY COST= $/HR        31.15

      COLUMN CAPITAL COST    = $     55225.00
1
 DISTILLATION COLUMN CALCULATION ... RIGOROUS MATRIX METHOD

 STREAM NAME:                    S1         S2         S3
                              LBMOL/HR   LBMOL/HR   LBMOL/HR
  1 *ETHANE                    30.0000    29.9998    0.00017
  2 *PROPANE                   200.000    191.963    8.03746
  3 *N-BUTANE                  370.000    4.03214    365.968
  4 *N-PENTANE                 350.000    0.00549    349.995
  5 *N-HEXANE                  50.0000    0.00000    50.0000
 TOTAL LBMOL/HR                1000.00    226.000    774.000
 TOTAL LB/HR                   60786.3    9601.18    51185.2
 1000 BTU/HR                   -772.83     269.50    -354.29
 DEGREES F                      225.00     118.28     265.32
 PSIA                          250.000    248.000    252.000
 DENSITY, LB/FT3                0.0000     2.2117    27.9166
 MOLE FRAC VAPOR                0.2295     1.0000     0.0000

1
 DISTILLATION COLUMN CALCULATION ... RIGOROUS MATRIX METHOD


 D1       - AFRAC -
        FEED 1 IS S1    ON STAGE  9
          2 -4.82E-01
  LAMBDA       1.93E-03  8.93E-07
         44 -2.48E-03
         48  6.03E-05
         49  6.11E-05
  LAMBDA       2.16E-03  1.97E-07
         81 -9.96E-07
    *C1     -ITER  1, MANIP PARAM= 6.8270E+00, CALC VAL= 6.6956E+00
             DESIRED VAL= 8.0000E+00, NEW MANIP PARAM= 6.1270E+00
 D1       - AFRAC -
        FEED 1 IS S1    ON STAGE  9
          2 -1.32E-01
  LAMBDA       2.16E-03  1.96E-07
         37  1.30E-03
         40 -2.91E-05
         41 -6.34E-06
         42  8.00E-07
```

```
   *C1     -ITER  2, MANIP PARAM= 6.1270E+00, CALC VAL= 7.8779E+00
             DESIRED VAL= 8.0000E+00, NEW MANIP PARAM= 6.0547E+00
D1      - AFRAC -
       FEED 1 IS S1   ON STAGE  9
         2 -1.37E-02
 LAMBDA      1.93E-03  3.13E-07
        34  2.21E-04
        36  5.10E-06
        37 -5.92E-07
  **C1     -CONVERGED IN ITER  3, MANIP PARAM= 6.0547E+00
CD1    (CAFRC)  COST FOR UNIT D1       CAP= 55225.$, UTL=  31.15$/HR
  **END OF HISTORY


***   End of Files for EX #4C   ***
```

APPENDIX G
PROCEDURE FILES FOR LOADING THE VARIOUS FORTRAN PROGRAMS

Program to set up PGM in the file PGM.EXE. Here PGM represents VLE, PROPTY, and PREPRO.

```
/LOGON
/REM
/REM ... FILE NAME "PGM.LOAD"
/REM     FILE GENERATES THE 'PGM.EXE' LOAD MODULE
/REM
/ERASE *
/STEP
/REM ... /PROC C,(&LNG,&SOURCE,&LST,&OBJ,&XRF,&MAP,&BUG)
/REM     &LST = LIST   = YES/NO
/REM     &OBJ = OBJECT = YES/NO
/REM     &XRF = XREF   = YES/NO
/REM     &MAP = MAP    = YES/NO
/REM     &BUG = DEBUG  = YES/NO
/DO COMPILE,($BGFOR,PGM.FOR,NO,YES,NO,NO,YES)
/DO COMPILE,($BGFOR,FT.CHAR.FOR,NO,YES,NO,NO,YES)
/DO COMPILE,($BGFOR,FT.TIMEDATE.FOR,NO,YES,NO,NO,YES)
/REM
/REM ... CONTINUATIONS IN $ASSEMB REQUIRE A NON BLANK IN
/REM     COLUMN 72, AND THE FOLLOWING LINES STARTING IN
/REM     COLUMN 16.
/REM === /PARAM ASMLST=YES
/PARAM ASMLST=NO
/EXEC $ASSEMB
 DDS
DSREF=3,RECFORM=FIXUNB,BLKSIZE=160,RECSIZE=160,X
               TYPEFLE=INOUT,DEVICE=CORE
 DDS
DSREF=8,RECFORM=FIXUNB,BLKSIZE=480,RECSIZE=480,X
               TYPEFLE=INOUT,DEVICE=CORE
 DVLST 1,2,5,6,7,97,98,99,3,8
 END
/EXEC $LMR
 CONTROL OUTFILE=(PGM.OML),LISTING=(MODNAMES,SYSLST)
  COPYALL SOURCE=*
 END
/EXEC $TSOSLNK
 PROG PGM,FILENAM=PGM.EXE,VERSION=66
  INCLUDE PGM,PGM.OML
  INCLUDE $BLOCK,PGM.OML
  INCLUDE ILF#DS3,PGM.OML
  RESOLVE ,PGM.OML
 BIND
 END
```

```
/STEP
/ERASE *
/STEP
/ERASE PGM.OML
/STEP
/REM
/LOGOFF
```

Program to generate the FLOWTRAN Subroutine Library used by the linkage editor to resolve all subprogram references.

```
/LOGON
/REM
/REM ... FILE NAME "FT.GEN.LIBRARY.LOAD"
/REM     FILE GENERATES THE FLOWTRAN SUBROUTINE LIBRARY
/REM
/ERASE *
/STEP
/REM ... /PROC C,(&LNG,&SOURCE,&LST,&OBJ,&XRF,&MAP,&BUG)
/REM     &LST = LIST    = YES/NO
/REM     &OBJ = OBJECT  = YES/NO
/REM     &XRF = XREF    = YES/NO
/REM     &MAP = MAP     = YES/NO
/REM     &BUG = DEBUG   = YES/NO
/DO COMPILE,($BGFOR,FT.FOR,NO,YES,NO,NO,YES)
/DO COMPILE,($BGFOR,FT.CHAR.FOR,NO,YES,NO,NO,YES)
/DO COMPILE,($BGFOR,FT.TIMEDATE.FOR,NO,YES,NO,NO,YES)
/REM
/REM ... CONTINUATIONS IN $ASSEMB REQUIRE A NON BLANK IN
/REM     COLUMN 72, AND THE FOLLOWING LINES STARTING IN
/REM     COLUMN 16.
/REM === /PARAM ASMLST=YES
/PARAM ASMLST=NO
/EXEC $ASSEMB
 DDS
DSREF=3,RECFORM=FIXUNB,BLKSIZE=160,RECSIZE=160,X
               TYPEFLE=INOUT,DEVICE=CORE
 DDS
DSREF=8,RECFORM=FIXUNB,BLKSIZE=480,RECSIZE=480,X
               TYPEFLE=INOUT,DEVICE=CORE
 DVLST 1,2,5,6,7,97,98,99,3,8
 END
/EXEC $LMR
 CONTROL OUTFILE=(FT.GEN.LIBRARY),LISTING=(MODNAMES,SYSLST)
  COPYALL SOURCE=*
 END
/ERASE *
/STEP
```

```
/REM
/LOGOFF
```

Procedure file to compile and link.

```
/REM === FILENAME: COMPILE
/REM ===
/REM
/REM
/REM === PROC "COMPILE"
/PROC C,(&LNG,&SOURCE,&LST,&OBJ,&XRF,&MAP,&BUG)
/PARAM LIST=&LST,OBJLST=&OBJ,XREF=&XRF,MAP=&MAP
/PARAM DEBUG=&BUG,ASMLST=&LST
/SYSFILE SYSDTA=&SOURCE
/EXEC &LNG
/SYSFILE SYSDTA=(PRIMARY)
/ENDPR
```

BIBLIOGRAPHY

Clark, J. Peter, Thomas P. Koehler, and Jude T. Sommerfeld. *Exercise in Process Simulation Using FLOWTRAN*. 2nd ed. Salt Lake City: CACHE, 1980.

Hill, Louis A., Jr. *Structured Programming in FORTRAN*. Englewood Cliffs: Prentice-Hall, 1981.

Seader, J. D., W. D. Seider, and A. C. Pauls. *FLOWTRAN Simulation -- An Introduction*. 2nd ed. Cambridge: CACHE, 1980.