Fault tree synthesis for real time and design applications on process plant

This item was submitted to Loughborough University’s Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/11843](https://dspace.lboro.ac.uk/2134/11843)

Publisher: © Gregorio Alberto Martin-Solis

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
LOAN C

Date due: - 4 FEB 1991
Loan 3 wks. + 3
Unless recalled

Date due: - 9 SEP 1991
Loan 3 wks. + 3
Unless recalled

Date due: - 12 JUN 1991
Loan 3 wks. + 3
Unless recalled

Date due: - 1 JUL 1994
## APPENDIX II

### RTL/2 AND RSX DESCRIPTION

#### Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>II.1 Introduction</td>
<td>340</td>
</tr>
<tr>
<td>II.2 RTL/2</td>
<td>341</td>
</tr>
<tr>
<td>II.2.1 Global structure</td>
<td>341</td>
</tr>
<tr>
<td>II.2.1.1 Multitasking</td>
<td>342</td>
</tr>
<tr>
<td>II.2.1.2 Re-entrant code</td>
<td>343</td>
</tr>
<tr>
<td>II.2.1.3 Supervisors</td>
<td>343</td>
</tr>
<tr>
<td>II.2.2 Text of a program</td>
<td>344</td>
</tr>
<tr>
<td>II.2.3 Data structure</td>
<td>346</td>
</tr>
<tr>
<td>II.2.3.1 Reference modes</td>
<td>346</td>
</tr>
<tr>
<td>II.2.3.2 Arrays</td>
<td>347</td>
</tr>
<tr>
<td>II.2.3.3 Records</td>
<td>349</td>
</tr>
<tr>
<td>II.2.4 Brick Declaration</td>
<td>354</td>
</tr>
<tr>
<td>II.2.5 Modules</td>
<td>355</td>
</tr>
<tr>
<td>II.2.6 Input/Output</td>
<td>357</td>
</tr>
<tr>
<td>II.3 RSX</td>
<td>358</td>
</tr>
<tr>
<td>II.3.1 RSX-11M tasks</td>
<td>359</td>
</tr>
<tr>
<td>II.3.1.1 Overlay structures</td>
<td>359</td>
</tr>
<tr>
<td>II.3.1.2 Task creation process</td>
<td>360</td>
</tr>
<tr>
<td>II.3.2 Service at peak load</td>
<td>360</td>
</tr>
<tr>
<td>II.3.3 The Executive</td>
<td>364</td>
</tr>
<tr>
<td>II.3.4 Utilities</td>
<td>365</td>
</tr>
<tr>
<td>II.4 Program Development</td>
<td>367</td>
</tr>
<tr>
<td>Figures: II.3.1 Sequential Program execution</td>
<td>373</td>
</tr>
<tr>
<td>II.3.2 Concurrent Program execution under RSX-11M</td>
<td>373</td>
</tr>
<tr>
<td>II.4.1 Steps involved in producing an RTL/2 written task for RSX-11M</td>
<td>374</td>
</tr>
<tr>
<td>II.4.2 Preparing a source program for execution</td>
<td>375</td>
</tr>
<tr>
<td>References</td>
<td>376</td>
</tr>
</tbody>
</table>
Appendix II

RTL/2 and RSX-11M Description

II.1 Introduction

The aim of this appendix is to lump together the information, scattered in several references, that was considered to be needed in order to have a better understanding of the programs and tasks used in this work.

It is not the function of this appendix to be a training manual for RTL/2 nor for the operating system RSX-11M. A complete description of the syntax of the language and the facilities of the operating system may be found in the references given at the end of this appendix.

Sections II.2 and II.3 of this appendix describe the main features of the language and of the operating system relevant to this work. Section II.4 describes in some detail the steps involved in producing a written task using RTL/2 as the language for the source files, and RSX-11M as the operating system by means of which the task is built. Some examples are used to illustrate the different steps.
II.2 RTL/2

RTL/2 is a high-level programming language developed at the Corporate Laboratory of ICI Ltd. during the period 1969-72. It is intended primarily for use in multitask systems on smaller computers. The language resulted from a broad research programme aimed at providing a basic software suitable for real time applications. RTL/2, although a new language and not a subset or variant of any existing language, clearly incorporates many features of other languages such as Algol 60, Algol 68, Algol W, BCPL, Coral 66, FORTRAN PL/1 and POP-2.

II.2.1 Global structure

Any RTL/2 program requires a certain minimum set of functions to be provided by the computer in order to run. These functions range from simple things such as fixed and floating-point arithmetic, to more complicated mechanisms such as procedure entry and exit, and array subscript checks. These functions are sometimes provided by hardware but most of the time the hardware will have to be enhanced by software routines. These routines are known as control routines and by their very nature cannot be written in RTL/2.

Apart from these routines and any similar software required for environmental purposes, the software in a computer programmed in RTL/2 is said to form a program complex. A program complex is a collection of items known as bricks. They are of three types:
1) **Procedure Brick.** This is a read only piece of coding describing an executable process. It may have parameters and local variables but the latter are restricted to be scalars (and not structures such as arrays). The coding of a procedure may directly access variables in a data brick, but not the local variables or parameters of another procedure. A procedure may not include internal procedures.

2) **Data Brick.** This is a named global static collection of scalars, arrays and records (similar to a common in FORTRAN).

3) **Stack Brick.** This is an area used as workspace by a task for the storage of local variables, links and other housekeeping items.

In addition to the three basic types of brick there are two special classes of brick which can be accessed by RTL/2 coding but not necessarily completely written in RTL/2. These are called SVC procedure and data brick. The RTL/2 text describing several bricks may be grouped together to form a module which is the unit of compilation.

II.2.1.1 **Multitasking**

A task, broadly speaking, is an identifiable execution of a logically coherent sequence of code and is to be distinguished from procedures which are merely passive. In other words, procedures indicate what is to be done whereas a task is the active doing. In a multitasking situation it is often convenient to have code which can be used by two or more tasks in parallel. This requires for each task to have a private work-space independent of other tasks. In RTL/2 this is done through the stack. Each task has its own stack which holds the values of local variables of procedures, subroutine links and other information relevant to the active execution of a
series of procedures. Procedures called as sub-routines could be private to a task or shared re-entrantly with other tasks. Data bricks thus provide permanent storage which can either be private to a task or act as a communication area between tasks. The latter case was used during this work, a general data brick (Data Base) was created to be shared by the different tasks described in Appendix III.

II.2.1.2 Re-entrant code

Re-entrant code is of considerable value in multitasking systems since it allows a single copy of common routines to be used by several tasks simultaneously. This saves on core space by removing the need for multiple copies of routines and clarifies the underlying structure of a system. I/O routines is a good example of this and are found in the resident library FTREES.RTL described in Appendix III.

II.2.1.3 Supervisors

In a multitasking system it is usually possible to consider the system as comprising at least two levels of software. The top level, usually known as the supervisor will itself be a program complex. It will usually be responsible for handling interrupts and scheduling the task of the complexes at lower levels. The SVC procedure used in RTL/2 is provided as a means of communication from the lower levels to the supervisor. Typical uses of SVC procedures would be for the creation, control and elimination of tasks, for the initiation of input/output, for the management of message transfers between task and so on. RTL/2 provides SVC procedures and SVC data bricks, but in
contrast to normal bricks SVC procedures and data bricks are outside RTL/2 itself since they are likely to be very system dependent. An SVC procedure can only be called and not used as a literal value and the variables in an SVC data brick cannot be used as initial values of reference variables in other data bricks.

II.2.2 Text of a program

The characters forming the text of an RTL/2 program are grouped together into basic items of various sorts and the text is then considered as a sequence of these items. The various basic items are:

1) **Names.** A name consists of the usual sequence of letters and digits of which the first is a letter. There is no limit to the number of characters in a name and all characters are significant. The actual character set chosen for the representation of RTL/2 is a subset of ISO7. It can be found in Ref. 15 of this appendix.

2) **Numbers.** Numbers are used to denote literal and initial values of the following four numerical modes:

   a) **BYTE.** The byte mode represents an unsigned integer in the range (0,255). It is primarily intended for character handling but can equally well be used for the manipulation of any small integers or flags. Its range is explicitly specified and so the BYTE mode is truly machine independent.

   b) **REAL.** The real mode is provided for normal floating-point arithmetic particularly on machines with appropriate hardware. This mode was not used in this work in order to save the limited space available.

   c) **INT.** The integer mode represents a signed integer. It assumes a minimum word length of 16 bits. The integer can also be considered to represent a bit pattern and logical operations can be performed on integers if they are considered in this way. This feature of the mode INT was used sometimes in this work.
d) **FRAC.** The fraction mode is used to overcome the problem of fixed point arithmetic due to machine dependence. A fraction value lies in the range \([-1, +1)\), that is including -1 but not +1. This mode was used to specify the probability values given to the basic events.

3) **Strings.** Strings are used to denote literal and initial values of arrays of bytes. They are most commonly used as parameters of procedures for the output of text. A string is basically a sequence of characters enclosed in inverted commas and denotes the set of values formed by taking the ISO7 values of those characters. For example "TEST" denotes the set of byte value 'T', 'E', 'S', 'T'.

4) **Comments.** In RTL/2 a comment is simply enclosed between a pair of % characters. A comment may appear virtually anywhere and may contain any characters except a new line. Comments were used at the top of each of the procedures described in Appendix III to give a brief explanation of each program's function.

5) **Titles.** A title provides a means of labelling the object code or a listing of the source code in whole or in part. It consists of the keyword TITLE followed by any sequence of characters and is terminated by a semicolon. Titles were used in this work to label each module of compilation.

6) **Options.** An option provides a means of informing the computer of any special style of compilation which may be required. It consists of the keyword OPTION followed by an identifying number in brackets and then a series of options separated by commas. A common option used in this work was Bound Check (BC). For a complete list and explanation of the options available see Ref. 3 of this appendix.

7) **Separators.** The remaining items are separators which are just the operators and punctuation symbols. They are grouped into simple separators consisting of a single character and compound separators consisting of several characters. Examples of compound separators are:

```
: = assignment
// = fraction division
> = greater than or equals.
```
Examples of simple separators are:

(  left bracket
+  plus
,  comma

the whole set of separators is described by Ref. 15 of this appendix.

8) LET replacement. LET is a simple non-parameterized replacement facility, that enables the user to give a name to a sequence of items and use this name instead of the sequence. This facility was widely used in the definition of the Data Base program shown in Appendix III.

**II.2.3 Data structures**

RTL/2 includes reference modes (REF) and two forms of compound structures; arrays and records.

**II.2.3.1 Reference modes**

Reference variables can take as value the addresses of variables of primitive modes (INT, REAL, BYTE, FRAC, etc.) or addresses of arrays or records. All references are constrained to point only one type of item and this is specified in the declaration. Thus:

```plaintext
INT I;
REF INT RI := I;
```

will declare and integer variable I and a reference variable RI constrained to take as values, the address of integers and which is initialised in this case to contain the address of I. This will look like:
A reference may be declared to be a reference to any mode except other references.

II.2.3.2 Arrays

RTL/2 allows arrays of any number of dimensions. An array is a compound structure consisting of an indexable set of components of the same mode. Arrays are declared as:

```
ARRAY(3) INT A;
```  

This indicates that A is an array of three integers, the elements of an array may be of any mode other than arrays themselves. References to arrays are declared as:

```
REF ARRAY INT RA := A;
```  

this declares a reference variable RA and initialises it to point the array A. Elements of an array are accessed by appending a subscript in round brackets to the array identifier (or to ref array variable). Thus A(2) denotes the second element of A. In the above example with RA initialised to A, RA(2) will give indirect access to the same
variable. The subscripts can be any expression which gives an integer value.

Initial values of arrays are denoted by a list of the values for the individual elements separated by commas and enclosed in brackets, as an example:

\[
\text{ARRAY(3) INT A := (4,7,10);}\
\]

If several successive elements are the same then a replica factor in brackets may be used. For example:

\[
\text{ARRAY(10) REAL A := (0.1,0.0(8),0.1);}\
\]

In the example \(A(1)\) and \(A(10)\) are initialised to \(0.1\) and the other elements to \(0.0\). Multidimensional arrays are handled as arrays of references to arrays. Consider the following declarations:

\[
\text{ARRAY(3) REF ARRAY INT P := (P1,P2,P3);}\
\]

\[
\text{ARRAY(4) INT P1,P2,P3;}\
\]

These declare an array \(P\) of length \(3\) whose elements are references to the arrays \(P1, P2, P3\). The arrays \(P1, P2, P3\) are all arrays of \(4\) integers. The structure created would look as follows:
Alternatively the declaration \texttt{ARRAY(3,4)INT P} will create the same structure except that the subarrays are now anonymous. Access to elements of \texttt{P} can be expressed as \texttt{P(I)(J)} or \texttt{P(I,J)} which are equivalent.

\textbf{II.2.3.3 Records}

A record is a data structure consisting of several components. A record belongs to a record class defined by a \texttt{MODE} definition which indicates the modes of the individual components and the identifiers by which the components may be selected. This is indeed a powerful feature of RTL/2 because it enables the programmer to declare new modes that will enable the convenient handling of groups of related information. This feature allows the use of the list processing techniques. A general discussion of list processing for simple problems is given by Foster.\textsuperscript{(5)}

As an example of the use of new modes declarations consider the
following:

MODE CELL (INT VA, REF CELL NEXT);

there are two components in this record, the first VA contains an integer value, whilst the second NEXT points to another record of the same class. Note that mode declaration effectively defines a new language word in this case CELL. Individual records, array of records and references to records can be declared using the word CELL as if it were a keyword such as INT, or REAL. Consider the following example:

CELL BASE;
REF CELL NEXTCELL;
ARRAY(100)CELL CELLS;

The declarations declare a single record BASE, a reference variable constrained to point to records of this class and an array of records. This will look like:

BASE  

NEXTCELL  

CELLS  

......
Initial values of records are indicated using a notation similar to that for arrays with the initial values of the components separated by commas and the whole in brackets. Unlike arrays repetition factors are not allowed. If the former example is initialised as follows:

\[
\begin{align*}
\text{CELL BASE: } &= (0, \text{CELLS}(1)); \\
\text{REF CELL NEXT CELL: } &= \text{CELLS}(2); \\
\text{ARRAY(100)CELL CELLS: } &= ((3, \text{CELLS}(2), (7, \text{BASE}), (0, \text{BASE})(98));
\end{align*}
\]

The interlinked structure will look as follows:

Record components are accessed by appending a point (.) followed by the component name to the record identified (or to the ref record variable). For the example being considered access to the record component will be as follows:
Mode definitions must be placed outside data bricks, and apply to the whole of the module. The only things that are not allowed as components are records and arrays of records; any other modes or ref modes (including ref records) or arrays of them are allowed.

In the example used to illustrate the manipulation of records there are two declarations BASE and NEXTCELL that can be used as "pointers" to manipulate the list of CELLS. The pointers are used to "remember" a place in the list and to modify this list. In the example mentioned BASE is the pointer used to point the top of the list and NEXTCELL is used to manipulate the list. A way to move down in the list and link CELL(2) to CELL(3) by means of the pointers of the example could be:

\[
\begin{align*}
\text{NEXTCELL} & := \text{CELLS}(3) \\
\text{CELLS}(2) \cdot \text{NEXT} & := \text{NEXTCELL}
\end{align*}
\]

The interlinked structure will look as follows:
Sometimes in order to terminate the list the last element should point to no other element of the list. To solve the problem NILCELL is defined as follows:

\[
\text{CELL NILCELL} := (0, \text{NILCELL})
\]

Note that the next element of NILCELL is pointing to itself. The list structure for the array CELLS can be set up with the help of the pointers BASE, NEXTCELL in the same way as for CELLS(2), but the last element will point now to NILCELL. This could be done as follows:

\[
\text{NEXTCELL} := \text{NILCELL} \\
\text{CELLS(100).NEXT} := \text{NEXTCELL}
\]

The interlinked structure will look now as a chain of elements. BASE will be pointing the first element and the last element will be pointing NEXTCELL.
Pointers are very useful in list processing and in this work several of them are used to build the trees and to "remember" places in the Data Base. All the different modes, its components, pointers and important variables of the programs used in this work are discussed in Appendix III.

II.2.4 **Brick declaration**

Two of the three kinds of bricks mentioned when the global structure was discussed are used in this work: Data bricks containing global static data and procedure bricks defining executable code.

1) **Data bricks.** The declaration of a data brick consists of the keyword DATA followed by its name and a semicolon, a series of declarations separated by semicolons and finally the keyword ENDDATA. Thus the examples used to illustrate records before, could be declared in a data brick with the name CELLDATA as follows:

   DATA CELL DATA
   CELL BASE;
   REF CELL NEXTCELL;
   ARRAY(100)CELL CELLS;
   ENDDATA;

2) **Procedure Bricks.** The declaration of a procedure brick commences with the keyword PROC followed by the name of the procedure, a description of its parameters and result (if any), a semicolon, a body describing the action of the procedure and finally the keyword ENDPROC. The description of the parameters is very similar to that of the components of a record. It consists of a list of modes and names separated by commas and enclosed in brackets. The brackets must always be present even if there are no parameters. Consider the following example:
PROC SUM(REF ARRAY REAL A) REAL;
  REAL T := 0.0;
  FOR I := 1 TO 10 DO
    T := T + A(I)
  REP;
  RETURN(T);
ENDPROC;

This example is a procedure SUM which takes a single parameter A of mode REF ARRAY REAL. The procedure defines a function as indicated by the keyword REAL following the bracket and preceding the semicolon on the first line. The procedure body consists of the declaration of the local variable T and initialises T to 0.0. There then follows a FOR statement which sums the elements of the array and finally a RETURN statement to pass the real result back to the expression from which the procedure is call.

Most of the programs shown in Appendix III are procedure bricks that use the data brick declared in the Data Base program DATBAS.RTL. A complete description of the statement structure and the operators available in RTL/2 can be found in references 15, 16, 17 and 18 of this appendix.

II.2.5 Modules

A module in RTL/2 is the unit of compilation. It consists basically of a collection of bricks, probably, but not necessarily related. If a brick declaration is preceded by the keyword ENT then its name will be globally known and the brick accessible from other modules. Conversely in order to access a brick in another module an adequate description of that brick must be given to the compiler so that the correct mechanisms can be compiled. In the case of an external procedure the number and modes of the parameters (and result)
have to be given but their names are immaterial. The description consists of the keywords EXT and PROC followed by the procedure description and a list of brick names. For example a module using the trigonometric functions SIN, COS, and TAN could express these as external as follows:

\[
\text{EXT PROC (REAL) REAL SIN, COS, TAN}
\]

these bricks have as parameter a real and will deliver a real.

In the case of data bricks the situation is a little bit different, the names of the internal variables have to be given so that they can be referenced individually. In fact the description of an external data brick is identical to its declaration preceded by EXT except that the variables must not be given initial values.

A module can contain:

1) Bricks to be compiled
2) LET definitions
3) Titles and Options
4) External descriptions
5) Record mode definitions.

A suitable order for a module could be:

1) OPTION statement
2) TITLE
3) LET Definitions
4) MODE Definitions
5) EXT procedure descriptions
6) EXT Data brick descriptions
7) Data brick declarations
8) Procedure declarations.
II.2.6 Input/Output

The way in which data may be transferred between a program and a physical device depends very much upon the characteristics of that device. It may handle whole records at a time or merely a stream of individual characters. In RTL/2 a standard has been set for stream-oriented I/O. The kernel of the mechanism is an SVC data brick containing two procedure variables. The SVC data brick is as follows:

SVC DATA RRSIO

PROC ( ) BYTE IN;

PROC ( BYTE ) OUT;

ENDDATA;

The procedure in IN (when called) removes the next character from the current input stream and returns it as a result. The procedure in OUT (when called) sends the character passed as parameter to the current output stream. Thus the channel that the current input or output stream manipulates is simply defined by the procedure that is the value of IN or OUT. A channel is not restricted to being a physical device, it might simply be an internal array. The switching of streams is carried out by assigning different literal procedures to the procedure variable IN and OUT. This can be done directly by the user or by the system on his behalf. This was carried out in the programs of this work frequently. (See Appendix III proc COMIP.TRL). A detailed description of the I/O facilities in RTL/2 based in the procs IN and OUT can be found in Ref. 16 of this appendix.
II.3 RSX

The RSX-llM Realtime Operating System can be used on any DEC PDP-11 computer, except PDP-11/03. The minimum configuration is a 16K PDP 11/10 with a teletype. The computer used in this work was a PDP 11/20 (28K). RSX-llM is designed for application requiring response to physical events as they occur. Real time operation is based on the continuous interaction of tasks stored within RSX-llM in response to events occurring in the outside world. The tasks or programs stored within RSX determine how the system will react to real-world events. Response may consist of selecting and sending control information back to the process being monitored as in a chemical manufacturing facility. It may involve a series of calculations as in laboratory data processing. In the case of an operator command entry, the response might involve starting the execution of a new task.

RSX-llM is an event-driven system, that is, the occurrence of a significant event causes the scheduling, interruption, execution, or resumption of programs or tasks that have been written to respond to that event. The events are presented to the RSX-llM Executive as interrupts which can be either internal or external to the system. In the case of external events, such as the completion of an analog-to-digital conversion, the interrupts are generated by hardware devices connected to the computer system and directed to the relevant device driver. In the case of internal events, such as the request of a task execution by another task, the interrupts are generated by software and recognised by the RSX-llM Executive. In both cases, the interrupts are channelled to the proper service routine by the hardware. The
combination of the PDP-llM hardware interrupt system and the RSX-llM software service routine provides the efficient interrupt processing required for effective real time operation.

II.3.1 RSX-llM tasks

Programs that have been fashioned into executable units are called tasks and the program which creates tasks is called the Task Builder. It is a vital element in the overall operation of a software system.

All tasks in RSX-llM are stored on disk in memory image format and are retrieved by name. A task image is produced by a linking operation performed by the Task Builder. It incorporates all the elements necessary to make a task ready for execution. Typically the Task Builder operates on the output of a compiler or assembler which are binary files. (In this work the compiler was the RTL/2 one). It integrates the main program and its subroutines into a single file and establishes the links with the common data areas and re-entrant library routines that the task needs to access.

II.3.1.1 Overlay structures

The memory resource of a computer is fixed and finite. If a program will not fit in the available memory it must be split up or overlaid. Overlaid tasks share the fixed memory such that when one part of the program is complete, it is overlaid by another. It is the function of the Task Builder to create, from a set of user overlay specifications, the overlaid task structure. For some of the tasks
used in this work it was necessary to use overlay structures; they can be found described in Appendix III.

II.3.1.2 Task creation process

The task creation process can be summarised in three steps. The user must:

1) Prepare a program section or sections in a supported language (RTL/2 in this work).
2) Submit each program section to the appropriate language translator.
3) Submit the translated program sections to the Task Builder.

The functions of the Task Builder can be summarised as follows:

1) Reallocates each object module and assigns it an absolute address.
2) Links the modules by correlating global symbols.
3) If the user so wishes, it prints a load map which displays the assigned absolute address.
4) Searches the disk-resident library of subroutines, and links subroutines containing global symbols requested by other object modules.
5) Outputs the final linked program that can be later loaded to the Executive.
6) Creates overlay segments to facilitate the execution of large programs.

III.3.2 Service at peak load

RSX-11M provides facilities for minimising absolute response to a single event and maximising the number of events that it can service at
peak load. Some of these facilities are:

1) **Multiprogramming**

The natural synchronisation requirements of programs coupled with the disparity between the time required for input/output transfers and the time required for processing can result in idle time for system resources. Multiprogramming is a method of maximising system efficiency by building a queue of demands for resources. The demand is achieved by maintaining in main memory two or more tasks waiting for a system resource. The concurrent tasks are then multiplexed, the needs of one task being serviced during the "dead time" intervals of the other task.

Multiprogramming is accomplished by dividing available memory into a number of named, fixed partitions. Tasks are built to execute out of a specific partition, and all partitions in the system can operate in parallel. If a task occupies the entire main partition, its operation is mutually exclusive with those tasks which occupy the area's subpartitions. The existence of subpartitions makes it possible to divide the space of a large partition among several smaller tasks. This is a very real requirement in a system which permits language translation concurrently with real-time processing, since language translators require large partitions and are used intermittently.

The advantages of multiprogramming can be explained by a simple illustration. Programs A, B and C are being executed in a system without multiprogramming. Program A reads some information from disk, operates on it, and displays a report. Program B performs some
computation, displays a message, performs some more computation, and writes the result to disk. Program C performs some computation, reads some information from disk, performs some more computation and writes the result to disk. Fig. II.3.1 illustrates the sequence in which various operations are performed as the three programs execute one after the other. Notice that while any part of the system such as a disk drive is being used, the other parts such as the central processor (CPU) are idle. Fig. II.3.2 shows the sequence in which the same functions are carried out under RSX-11M. Note that the three resources involved (CPU, disk drive, printer) are, at times, used simultaneously. Note also that concurrent execution of three programs require less computer time than sequential execution of the same programs.

2) **Priority scheduling**

Precise priority determinations assures accurate decisions on what task to run next. RSX-11M supports up to 250 priority levels, which may be assigned and amended by the operator as necessary. Tasks may be initiated by the operator or other tasks depending on the nature of the data to be acquired or the process to be controlled. Real time tasks execute in direct response to external interrupts, periodically (as in the scanning of analog/digital inputs), or at specified times during the day. Commands are straightforward and easy to use. For a complete description of the different commands see Ref. 11.

3) **Multitasking**

Multitasking is the multiprogramming of two or more tasks which
need to communicate among themselves and synchronise their activities. In a uniprocessor environment some response time dependencies can only be achieved by inducing parallelism. For example, in an airline reservation system a single console request may require several independent file accesses, and to achieve the needed response time, these accesses and their subsequent processing must occur in parallel under the control of a single master task. To exercise this control, the master task must be able to start, stop and synchronise activity with related subsidiary tasks. In effect, a multitasking capability gives every user task the ability to become an executive and thus extends the benefits of parallel execution to the user tasks. RSX-llM provides the services necessary to support multitasking.

4) Disk based operation

In many real-time applications, the total size of the programs or tasks that are necessary to perform a given job is larger than the size of the available computer memory. Also many of the tasks required to carry out the applications are executed in sequence rather than concurrently. Tasks which are not currently active are stored in secondary memory (disk) and quickly loaded into primary memory when needed. This makes it possible for the RSX-llM system to serve applications requiring fast response to real world events while efficiently handling large amounts of data.

In program development mode the disk is used to store programs, such as compilers, assemblers, editors and linkers as well as the files produced by the output of these system programs. Disk also provide a
convenient medium on which to store user data acquired by real time tasks before it is processed by analysis tasks to produce reports, statistical history and so forth.

RSX-llM maintains a copy on disk of all the tasks that have been installed in the system. These tasks are registered by name in disk directories, and by name and disk address in core-resident system tables. By registering installed tasks in primary memory, loading from disk involves only one simple fast disk transfer; this is a very significant feature for real-time applications where response is critical. Data files are also stored on disk referenced by name in a directory-structured organisation. These data files can be accessed randomly or sequentially by user programs.

5) Checkpointing

Temporary copies of tasks that have been "rolled-out" by the system are stored on disk. RSX-llM provides a checkpointing option (roll-in/roll-out) whereby low-priority tasks can be interrupted by high-priority tasks, copied onto the disk to make memory available to the high-priority tasks requesting execution and later resumed from the point of interruption.

II.3.3 The Executive

The RSX-llM Executive is permanently memory-resident and constitutes the heart of the operating system. The Executive provides the management facilities to allocate system resources to user and system tasks and to
arbitrate the conflicts that may arise between several tasks competing for the same resources it provides the services mentioned in Section III.3.2. A complete description of the Executive and its services can be found in the Executive Reference Manual. 

II.3.4 Utilities

RSX-11M provides the user with a comprehensive set of utility programs. The utility programs and their identifiers are as follows:

1) **Peripheral Interchange Utility Program (PIP)**. PIP is a file transfer program that provides the user with facilities for copying, renaming, listing, deleting and unlocking files.

2) **File Transfer Utility program (FLX)**. FLX is a file conversion program that provides the user with a facility to convert files between DOS-11 and RSX-11M formats and vice versa.

3) **File Dump Utility Program (DMP)**. DMP is a file listing program that provides the user with a facility for obtaining a printed copy of the contents of files.

4) **Line Text Editor Utility (EDI)**. EDI is an interactive context editing program that provides the user with a facility for creating and maintaining text files.

5) **Source Language Input Utility Program (SLP)**. SLP is a batch-oriented editing program that provides the user with a facility for creating and maintaining text files on disk.

6) **Librarian Utility Program (LBR)**. LBR is a library maintenance program that provides the user with a facility for creating, modifying, updating, listing, and maintaining library files.

7) **File Structure Verification Utility Program (VFY)**. VFY is a disk verification program that provides the user with a facility for verifying the consistency and validity of the file structure on a specified device.

Several of these utilities were widely used during the development and
testing of programs for this work. A detailed description of the utilities and their options can be found in the Utilities Procedures Manual for RSX-11M. (14)
II.4 Program Development

There are a number of steps that are involved in producing an RTL/2 written task for RSX-11M. These are:

1) Preparation of the source file.
2) Compilation to produce a macro file.
3) Assembly to produce a binary file.
4) Linking with other modules to produce a task image file.

These steps are illustrated in Fig. II.4.1, the whole process for preparing a source program for execution is illustrated in Fig. II.4.2. To illustrate the sequence of events involved in Fig. II.4.1 consider the following explanation of each point:

1) Preparation of source file

The source file should have the file extension "RTL" for convenient use of the compiler. Note that all modules presented to the compiler should have at least one OPTION statement and a TITLE. The text should follow the rules of syntax of the language RTL/2.

2) Compilation

A typical compiler command string would be:

```
RTL XAMPLE, TI:=XAMPLE/TI:XAMPLE/CS:XAMPLE
```

for a single source file (XAMPLE.RTL) to be compiled. The elements of the command are:
(1) The three letter command to RSX-11M.

(2) The compiler output file (XAMPLE.MAC).

(3) The listing output file. In the example a full listing will be produced on the initiating terminal.

(4) The source file (XAMPLE.RTL).

(5) The module TITLE.

(6) The CSECT name for use in the task builder.

For a short listing with two source files as was the case for most of the programs of this work the command might be (STDAT.RTL is the standard declarations file used in this work. See Appendix III for the listing).

```
RTL XXX,TI:/SS=STDAT,XXX/TI:XXX/CS:XXX
```

Full details of the compiler features, the errors produced during compilation and support facility are given in the RTL/2-RSX-11M manual. (3)

(3) **Assembly**

The compiler output is assembled using the MACRO assembler task. The command string for the example being considered would be:

```
MAC XAMPLE:=XAMPLE
```

where the elements of the command are:

(1) The three letter RSX-11M command.

(2) XAMPLE.OBJ binary output file.

(3) XAMPLE.MAC input file.
After assembly the compiler output file can usually be deleted using PIP.

```
PIP XAMPLE.MAC;*/DE
```

4) **Linking**

The linking process provides a large number of options to the user. A detailed description of them can be found in the Task Builder Reference Manual. To simplify matters it is assumed in this example that the linked output is destined by an RSX-11M system and not some other environment. For the example being considered the user creates the command file XAMPLE.CMD using EDI and containing the following task-builder command lines:

```
XAMPLE, TI:/SH=B1, XAMPLE       (1)
IOR, AR2, CTL, BA2, SLB/LB, ELB/LB (2)
/                                      (3)
STACK=200                               (4)
UNITS=3                                 (5)
ACTFIL=2                               (6)
ASG=TI:1:2:3                            (7)
PAR=GEN:40000:40000                   (8)
PRI=82                                  (9)
TASK=...XAM                            (10)
UIC=22,5                               (11)
//                                     (12)
```

A command file is used in order that the user does not have to re-type
the entire 12 lines if he modifies the task at a later date. The command lines have the following significance:

Line (1) specifies that:

a) The output file is to be called XAMPLE.TSK

b) A short listing of the task map is to be produced on the initiating terminal

c) The input files are to be BAL.OBJ, and XAMPLE.OBJ.

Line (2) specifies further input files IOR.OBJ, AR2.OBJ, CTL.OBJ, SLB.OLB and ELB.OLB. The last two files are object library files indicating that only modules required by the other file will be included in the task image output. A more detailed description of these files is given in the RTL/2-RSX-llM manual.

Line (3) specifies that no more files are to be built into the task. Subsequent lines specify task builder options. The options allow the user to tailor the task image to his exact requirements. Line (4) specifies the stack length. Line (5) specifies that 3 logical units (i.e. input/output devices) are to be used.

Line (6) specifies that only 2 of the logical units will require filing system support (Active files). Line (7) assigns the logical units to all the initiating terminal. Line (8) specifies that partition GEN is to be used and also gives its base address and length (in octal).

Line (9) specifies that the task priority is to be
82 (on a 1-250 range). Line (10) specifies the three letter task name to be XAM. The three dots are compulsory if the task is to be executed in the same manner as the system utilities EDI, MAC, PIP etc. This was the way used for the tasks used in this work to generate the fault trees and their names were restricted for this reason to three letters. Line (12) specifies the end of options input for the task builder. Note that there is no restriction that tasks must be linked (or compiled or assembled) on the "target" system on which the task is to run, normally another system will be used.

For real-time applications tasks must normally interact with each other via resident data areas. For economy reasons tasks may also share copies of common routines. Since RTL/2 code is re-entrant any sharing is completely transparent to the user; the only effect of linking to a shared area is to reduce the size of the user task. Data areas can obviously only be shared for communications purposes. Suites of user tasks on the PDP-11 of the Chemical Engineering Department used in this work will normally link up to 3 data or code shared regions:

1) The BA2, CTL, SLB shared region (FTREES for this work).
2) The plant I/O region (DATBAS in this work).
3) The user area.

An example of a task CAL used in this work, that is linked to some of these areas is given below in command file form:
This task has "read-only" access to the library of code in FTREES. The task has "read-write" access to the common data area in DATBAS. Note that in second line of the file includes the library T created for the different programs used in this work. All the command files used in this work to help in the building of the different task are shown in Appendix III.

Details of the RTL/2-RSX-11M facilities at the Chemical Engineering PDP-11/20 Computer are contained in Ref. 1 of this appendix.
Fig. III.3.1  Sequential program execution

Fig. II.3.2  Concurrent program execution under RSX-11M
Fig. II.4.1 Steps involved in producing an RTL/2-written task for RSX-llM
Fig. II.4.2 Preparing a source program for Execution
References


RSX-llM Manuals:

(6) "Executive Reference Manual"; (DEC-11-OMERA-B-D); Digital Equipment Corporation; Maynard Mass; 1975.

(7) "Introduction to RSX-llM"; (DEC-11-OMIEA-A-D); Digital Equipment Corporation; Maynard Mass; 1974.

(8) "I/O Drivers Reference Manual"; (DEC-11-OMDRA-B-D); Digital Equipment Corporation; Maynard Mass; 1974.


(10) "Macro-11 Reference Manual"; (DEC-11-OMMMAA-B-D); Digital Equipment Corporation; Maynard Mass; 1975.

(11) "Operator's Procedures Manual"; (DEC-11-OMOGA-B-D); Digital Equipment Corporation; Maynard Mass; 1975.

(12) "System Generation Manual"; (DEC-11-OMGIA-B-D); Digital Equipment Corporation; Maynard Mass; 1975.


(14) "Utilities Procedures Manual"; (DEC-11-OMUPA-B-D); Digital Equipment Corporation; Maynard Mass; 1975.
RTL/2 Manuals


(17) "System Standards"; RTL/2 ref. 4; ICI Ltd., 1974.

(18) "Training Manual"; RTL/2 ref. 3; ICI Ltd., 1974.
## APPENDIX III

### Programs and Tasks

#### Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>III.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>III.2</td>
<td>Records and the Data Base</td>
</tr>
<tr>
<td>III.2.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>III.2.2</td>
<td>Records</td>
</tr>
<tr>
<td>III.2.2.1</td>
<td>MODE BOUNDECO</td>
</tr>
<tr>
<td>III.2.2.2</td>
<td>MODE DATAV</td>
</tr>
<tr>
<td>III.2.2.3</td>
<td>MODE EVENT</td>
</tr>
<tr>
<td>III.2.2.4</td>
<td>MODE FAS</td>
</tr>
<tr>
<td>III.2.2.5</td>
<td>MODE GATE</td>
</tr>
<tr>
<td>III.2.2.6</td>
<td>MODE INDEX</td>
</tr>
<tr>
<td>III.2.2.7</td>
<td>MODE STREAM</td>
</tr>
<tr>
<td>III.2.2.8</td>
<td>MODE UNITDE</td>
</tr>
<tr>
<td>III.2.2.9</td>
<td>MODE VARIAB</td>
</tr>
<tr>
<td>III.2.3</td>
<td>Data Base</td>
</tr>
<tr>
<td>III.3</td>
<td>Tasks</td>
</tr>
<tr>
<td>III.4</td>
<td>Programs</td>
</tr>
<tr>
<td>Tables</td>
<td>III.1 Look-up Table for Faults</td>
</tr>
<tr>
<td></td>
<td>III.2 Look up Table for Type of Units</td>
</tr>
<tr>
<td></td>
<td>III.3 MODE BOUNDECO Description</td>
</tr>
<tr>
<td></td>
<td>III.4 MODE DATAV Description</td>
</tr>
<tr>
<td></td>
<td>III.5 MODE EVENT Description</td>
</tr>
<tr>
<td></td>
<td>III.6 MODE FAS Description</td>
</tr>
<tr>
<td></td>
<td>III.7 MODE GATE Description</td>
</tr>
<tr>
<td></td>
<td>III.8 MODE INDEX Description</td>
</tr>
<tr>
<td></td>
<td>III.9 MODE STREAM Description</td>
</tr>
<tr>
<td></td>
<td>III.10 MODE VARIAB Description</td>
</tr>
<tr>
<td></td>
<td>III.11 Description of the Pointers Defined in the Data Base</td>
</tr>
<tr>
<td></td>
<td>III.12 Description of the Tasks</td>
</tr>
<tr>
<td>Figures</td>
<td>III.2.3.1 Data base program</td>
</tr>
<tr>
<td></td>
<td>III.2.3.2 Memory layout used in this work</td>
</tr>
<tr>
<td></td>
<td>III.2.3.3 List structure for topology of a Two pipe and valve system</td>
</tr>
<tr>
<td></td>
<td>III.3.1 Task BMT indirect command file</td>
</tr>
<tr>
<td></td>
<td>III.3.2 Task BTR indirect command file</td>
</tr>
<tr>
<td></td>
<td>III.3.3 Task BTR overlay description</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>III.3.4 Task CAL indirect command file...</td>
<td>570</td>
</tr>
<tr>
<td>III.3.5 Task DEB indirect command file...</td>
<td>571</td>
</tr>
<tr>
<td>III.3.6 Task DEB overlay description...</td>
<td>572</td>
</tr>
<tr>
<td>III.3.7 Task DES indirect command file...</td>
<td>573</td>
</tr>
<tr>
<td>III.3.8 Task LIT indirect command file...</td>
<td>574</td>
</tr>
<tr>
<td>III.3.9 Task PMT indirect command file...</td>
<td>575</td>
</tr>
<tr>
<td>III.3.10 Task PRI indirect command file...</td>
<td>576</td>
</tr>
<tr>
<td>III.3.11 Task PTR indirect command file...</td>
<td>577</td>
</tr>
<tr>
<td>III.3.12 Task REU indirect command file...</td>
<td>578</td>
</tr>
<tr>
<td>III.3.13 Task RFA indirect command file...</td>
<td>579</td>
</tr>
<tr>
<td>III.3.14 Task RTU indirect command file...</td>
<td>580</td>
</tr>
<tr>
<td>III.3.15 Task RVA indirect command file...</td>
<td>581</td>
</tr>
</tbody>
</table>

### Listings

<table>
<thead>
<tr>
<th>Listings</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>III.1    Input Data for Pipe's Minitrees (Task BMT)...</td>
<td>582</td>
</tr>
<tr>
<td>III.2    Input Data for Heat Exchanger Minitrees (Task BMT)...</td>
<td>590</td>
</tr>
<tr>
<td>III.3    Input Data for a Two Pipe and Valve System (Task REU)...</td>
<td>609</td>
</tr>
<tr>
<td>III.4    Input Data for a Two Pipe and Valve System (Task DES)...</td>
<td>611</td>
</tr>
<tr>
<td>III.5    Output of a Two Pipe and Valve System (Task DEB)...</td>
<td>617</td>
</tr>
</tbody>
</table>
Appendix III

Programs and Tasks

III.1 Introduction

This appendix presents all the programs and tasks used to implement the fault tree synthesis methodology. Section III.2 describes the different MODES used and the structure of the data base. Section III.3 presents the different tasks used in this work and their options. Section III.4 provides a listing in alphabetical order of all the programs used by the different tasks described in Section III.3.

Listings of some of the output provided by the most used tasks are included in a special section at the end of this appendix.

III.2 Records and the Data Base

III.2.1 Introduction

One of the most useful features of RTL/2 during the development of the computer programs for this work, was the possibility of declaring new MODES. This feature was a powerful tool that allowed the author the freedom of action needed to manipulate the construction of fault trees using list processing techniques.

This section is devoted to describing the records that were declared for this work and the role played by them in the structure of the Data Base. In order to make the best use of the space available in the computer, the following approach was used in the definition of the
MODES whenever it was possible:

1) Use of 8 bit BYTES (rather than 16 bit INT's) to conserve storage space.

2) Use of bit manipulation to pack most of the information into BYTE or INT data.

3) Use of integers related to look-up tables instead of large repetitive strings.

4) Versatile records were preferred to specific ones.

The names given to each new MODE were related directly to its function. The look-up tables used for the faults and unit names are shown in Table III.1 and III.2. An example of the interlinked structure using the MODES described is also shown in this section.

III.2.2 Records

A description of the records used in the computer programs is presented in this section. The arrows from the boxes, are used to indicate in the figures, that those components are references to other records. An asterisk is used to indicate the fields that use bit manipulation to handle the data. Note that the spare field, that appears in some of the MODES, does not waste space.

III.2.2.1 MODE BOUNDECO

![Diagram of VARBC, NA1, NA2, NA3, NEXTBC with arrows pointing downwards]
This mode has the information related to the Boundary Conditions and Not-allowed faults of each event. Each field is described in Table III.3. Note that in this work the maximum number of faults used was 45. The number may easily be increased if needed, by adding new integers to the MODE declaration, i.e. NA4, NA5, etc.

III.2.2.2 MODE DATAV

This mode is used to store the data related to the measured variables. Its fields are described in Table III.4.

III.2.2.3 MODE EVENT

This mode is used in the description of each event that forms the trees and minitrees. Some of its fields are used in the construction of the minitrees only and others in the construction of the trees only. Its fields are described in Table III.5.
III.2.2.4  **MODE FAS**

This mode is used to deal with long strings in the programs such as the name of the units or the name of faults. It is an array of bytes and has only one field $L$.

III.2.2.5  **MODE GATE**

This mode is used to deal with the data related to the gates used in the trees and minitrees. The description of its fields is given in Table III.6

III.2.2.6  **MODE INDEX**

This mode is used to ease the location of the right minitree when the fault tree is being constructed. Its fields are described in Table III.7.
III.2.2.7 MODE STREAM

This mode is used to deal with the data of the unit's streams in the topology of the system under study. Its fields are described in Table III.8.

III.2.2.8 MODE UNITDE

This mode deals with the data related to each unit that forms part of the system under study. Table III.9 shows a description of its fields.

III.2.2.9 MODE VARIAB

This mode is used to deal with the data of the variables in a system. Table III.10 shows a description of its fields.
III.2.3 Data Base

With the help of the records described in the former section a Data Base was created. Fig. III.2.3.1 shows the actual program DATBAS.RTL that was used. It shows the restrictions of the Data Base due to the space limitations imposed by the hardware. Fig. III.2.3.2 shows the memory layout of the core image on the disk that was used in this work.

The Data Base shown in Fig. III.2.3.1 has several embodied comments that help to explain its structure. Several NIL declarations were needed and are also shown in Fig. III.2.3.1 the NIL declarations were needed because RTL/2 does not have an equivalent to the NIL feature of ALGOL-68. A brief description of the most important pointers defined in the Data Base is presented in Table III.11. Some of these variables are used at each stage of the fault tree development by the programs.

Fig. III.2.3.3 shows an example of the list structure for the topology of the Two Pipe and Valve system mentioned in Chapter 4, (Fig. 4.3). Note how the different elements of the arrays are interlinked to form the structure required by the algorithm to carry out the fault tree construction.

III.3 Tasks

Some of the tasks used in this work have already been mentioned elsewhere in this work. A brief description of each one of the tasks that have been used in this work together with their options is shown
in Table III.12. Task DEB is the only one with several possible combinations of options. The first part of the option before the second asterisk, refers to the number of the arrays that the user wants to print. The array options for this first part are:

1 - Events
2 - Gates
3 - Boundary Conditions
4 - First Main event of each unit
5 - Units
6 - Streams
7 - Variables
8 - Data for measured variables
9 - Faults
10 - Types of units
11 - Print all the arrays

The second part, after the second asterisk, refers to the part of the array stated in the first part option that the user wants to print.

The options for the second part are:

1 - Print the whole array
2 - Print the part of the array related to the minitrees
3 - Print the part of the array related to the trees.

These three options only apply to the first three arrays mentioned in the list of array options. For all the rest the second number should always be 1. A command line to use DEB could be as follows:
DEB TI := TI:*1*3

In this example the user would like to print the part of the array of events that is related to the tree that has been developed. Examples of the output produced by DEB are shown in listing III.5.

All the tasks except BTR can read/write data from/to the disk (DK$:) or the teletype. BTR can only read/write data from/to the teletype because (in order to save space) different I/O procedures were used. The output printed by BTR during the construction of the fault trees refers to warning or error messages. The indirect command files used to build the tasks together with their overlay description, where applicable, are shown in Figs. III.3.1 to III.3.15.

Examples of the use of each task are as follows:

```
BMT   DK$: A.TNT;1 = DK$: PIPE.DAT
BTR   TI: = TI:*1
CAL   TI: = TI:*0
DEB   DK$: VALVE.PRO;1 = TI:*3*3
DES   DK$: B.TNT;1 = DK$: HEXVAL.DAT
LIT   TI: = TI:
PMT   TI: = TI:
PRI   DK$: HEXVAL.PRO;1 = TI:
PTR   TI: = TI:
REU   TI: = TI:
RFA   TI: = DK$: FAULTS.DAT
TRU   TI: = DK$: TYPE.DAT
RVA   TI: = TI:
```
III.4 Programs

All the programs used by the different tasks mentioned in the previous section are included here in alphabetical order. Each program has a title or some comments at the top that gives a brief description of its function. The name of each program is restricted to six characters. Each name was given according to the function of each program, e.g. the program that deals with the garbage collection was called GARCOL.RTL etc.
% ALTERNATIVE INPUT PROCEDURE %

LET BUFLEN = 72;
LET IOSERR = 5840;
LET IPUT = OCT 1000;
LET QIOERR = 5838;
LET WAERR = 5839;

MODE BUFBLK (INT BDUMWRD,
    REF BYTE BUFBADD,
    INT BUFFSIZ, BUFSIZ, ESP1, ESP2, ESP3);

MODE IOSTAT (BYTE IOSTLOW, IOSTHIGH,
    INT IOSTVAL);

SVC DATA RRERR;
    LABEL ERL;
    INT ERN;
    PROC (INT) ERP;
ENDDATA;

DATA QIODAT;
    ARRAY(BUFLEN)BYTE BUF := (SF(BUFLEN));
    INT INEXT := 0;
    IOSTAT IS := ("8", "8", 2);
    BUFBLK BE := (12, BUF(1), BUFLEN, 0, 0, 0, 2);
    INT ITOTAL := 0;
ENDDATA;

EXT PROC () RRNUL;
EXT PROC (INT) RSXWTS, TESDSW;
EXT PROC (INT, INT, INT, INT, IOSTAT, PROC(), REF BUFBLK) RSXQIO;

ENT PROC ALMIP () BYTE;
    INEXT := INEXT + 1;
    FOREVER DO  % BUT ONLY IF USER INPUTS <CR> %
        IF INEXT < ITOTAL
            THEN RETURN(BUF(INEXT));
        ELSEIF INEXT = ITOTAL
            THEN RETURN(LF);
        ELSE RSXQIO(IPUT, 2, 2, 0, 1S, RRNUL, BB);
            TESDSW(QIOERR);

            RSXWTS(2);
            TESDSW(WAERR);

            IF I$1$IOSTLOW # 1
                THEN ERP(IOSERR);
            END;

            ITOTAL := I$1$IOSTVAL + 1;
            INEXT := 1;
        END;
    REP;

    RETURN(SP);  % JUST TO SATISFY COMPILER %
ENDPROC;
% ALTERNATIVE OUTPUT PROCEDURE %

LET BUFLN = 72;
LET IOSERR = 5017;
LET OPUT = OCT 480;
LET VFC = OCT 8;
LET QIOERR = 5016;
LET WAIERR = 5026;

MODE BUFLK (INT BUFDWDR,
    REF BYTE BBUFADD,
    INT BUFSIZE, BUFCHAR, BSP1, BSP2, BSP3);

MODE IOSTAT (BYTE IOSTLOW, IOSTHIGH,
    INT IOSTVAL);

SVC DATA RRERR;
    LABEL ERL;
    INT ERN;
    PROC (INT) ERP;
ENDDATA;

DATA QIODAT;
    ARRAY (BUFLN)BYTE BUF := (SP(BUFLN));
    INT ONEXT := 1;
    IOSTAT IS := ('B', 'O', '2');
    BUFLK BB := (12, BUF(1), I, VFC, 'E', 'B', 'O');
ENDDATA;

EXT PROC () RRNUL;
EXT PROC (INT) RSXWTS, TESDSW;
EXT PROC (INT, INT, INT, INT, INT, INT, IOSTAT, PROC()), REF BUFLK) RSXQIO;

EXT PROC ALMOP (BYTE B);
    BUF(ONEXT) := B;

    IF ONEXT = BUFLN OR B = CR OR B = VT
    THEN BUFSIZE := IF B = VT
        THEN ONEXT := 1
        ELSE ONEXT 
    END;

    RSXQIO(0PUT, 2, 2, IS, RRNUL, BB);
    TESDSW(QIOERR);

    RSXWTS(2);
    TESDSW(WAIERR);

    IF IS, IOSTLOW # 1
    THEN ERP(IOSERR)
    END;

    ONEXT := ONEXT + 1;
    RETURN;
ENDPROC;
% DATA OF A 'B' EVENT %

EXT PROC (REF FRAC, FRAC, INT, RAB, LABEL) GETFRA;

EXT PROC BEVT (REF EVENT E);
  E.BACKG := GATEPO;
  L1 := GETFRA(E.PROB, 0, 0, 1, "PROBABILITY", L1);
ENDPROC;
% BUILD MINITREES %

EXT PROC (REF EVENT) BEVT, MEVT, REVT, TEVT;
EXT PROC (REF INT, INT, INT, INT, RAB, LABEL) GETINT;
EXT PROC (REF BYTE, INT, RAB) GETBYT;
EXT PROC () REF EVENT NEXTFE;
EXT PROC (RAB) INT REFATR;
EXT PROC (REF BYTE, INT) SBITY;
EXT PROC () INT RETYUN;

EXT PROC BMINT();
    INT UN;
    BYTE TYEV;
    INT I, IN, TOTEV := 0;
    UN := RETYUN();
    WHILE UN $= 0 DO
        I := CI;
        GETBYT(TYEV, 3, "TYPE OF EVENT(* TO TERMINATE INPUT)");
        WHILE Tyev $= '*' DO
            BLOCK
                REF EVENT E := NEXTFE();
                E*UNID := BYTE (UN);
                E.IDEN := TYEV;
                GETBYT(E*VAR, 1, "VARIABLE");
                L1: GETINT(IN, 3, 1; "0-OUT, 1-1IN, 2-NO VAR, 3=1-VAR", L1);
                SBITY(E*INOU, IN);
                E*FAULT := REFATR("++");
                TOTEV := TOTEV + 1;
                IF Tyev = 'M'
                    THEN EVT(ITE).TOP := E;
                    END;
                    ITE := I;
                    MEVT(E);
                ELSEIF Tyev = 'T' THEN TEVT(E);
                ELSEIF Tyev = 'R' THEN REVT(E);
                ELSEIF Tyev = 'B' THEN BEVT(E);
                ELSEIF Tyev = 'B' THEN REVT(E);
                ELSE TWRT("*NL*EVENT UNDEFINED"
                    "(BMINT. RTL)");
                END;
                I := I+1;
            END_BLOCK;
            GETBYT(TYEV, 3, "TYPE OF EVENT(* TO TERMINATE INPUT)");
        END;
    ENDPROC;

PAGE
OPTION(1) EC, TR;

TITLE
BUILD MINITREES;

LET COMERR = 5000;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC() BMINT;
EXT PROC() RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOP, RSXOF;
EXT PROC(INT) INT RSXSWI, RSXSWO;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE ECHOIN, COMIP;

SVC DATA RRERR;
LABEL ER;
INT ERN;
PROC(INT) ERP;
ENDDATA;

SVC DATA RRSIO;
PROC () BYTE IN;
PROC (BYTE) OUT;
ENDDATA;

SVC DATA RRERRX;
INT LINDEX;
BYTE UFLAG, ERRLIN;
INT RSXDSW;
ENDDATA;

DATA FILLDAT;
ARRAY(70) BYTE FILE := "#SP(70)#";
ENDDATA;

ENT PROC RRJOB();
ERRLIN := 3;
RSXINT();
IN := COMIP;
IF IN() ≠ 'B' OR IN() ≠ 'M' OR IN() ≠ 'T' OR IN() ≠ ' ' THEN ERP(COMERR);
END;

TREAD(FILE, "=");
RSXOF(1, FILE);
RSXSWO(1);
FOR J := 1 TO LENGTH FILE DO
   FILE(J) := SP;
   REP;

TREAD(FILE, "*LF*"),
IN := ECHOIN;
RSXOP(2, FILE);
RSXSW(2);
BMINT();
RSXCL(1);
RSXCL(2);
OPTION(1) BC,TR;
TITLE
CONSTRUCTION OF TREE;

LET COMERR = 5000;
LET SP = OCT 42;
LET LF = OCT 12;

EXT PROC(INT) BUILTT;
EXT PROC (REF ARRAY BYTE,REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE ALMIP,COMIP;
EXT PROC (BYTE) ALMOP;
EXT PROC () INT IREAD;

SVC DATA RRERR:
 LABEL ERR;
 INT ERN;
 PROC(INT) ERF;
 ENDDATA;

SVC DATA RRSIO:
 PROC () BYTE IN;
 PROC (BYTE) OUT;
 ENDDATA;

SVC DATA RRXX:
 INT LINENO;
 BYTE UEFLAG,ERLUN;
 INT RSXDSW;
 ENDDATA;

DATA FILDAT:
 ARRAY (35) BYTE IFILE := "#SP(35)#";
 ARRAY (35) BYTE OFILE := "#SP(35)#";
 ENDDATA;

ENT PROC RRJOB();
 INT I;
 ERLUN := 3;
 IN := COMIP;
 IF IN() # 'B' OR IN() # 'T' OR IN() # 'R' OR IN() # '
   THEN ERP(COMERR);
END;

TREAD(OFILE,="");
TREAD(IFILE,="");
I := IREAD();
IN := ALMIP;
OUT := ALMOP;
BUILTT(I);
ENDPROC;
I BUILDING THE EVENTS

EXT PROC () REF VARIAB CHEVAR;
EXT PROC () CHEEDA, CHECKD, NEWEVT;

ENT PROC BUILDE();
IF FLAG = 0
THEN EST := EST.NEXT;
ELSE FLAG := 0;
END;

BLOCK
REF VARIAB VI := CHEVAR();
IF VI := NCVAN AND SUCH VARIABLE OR END
THEN CHECKD();
RETURN;
END;
VST := VI;
IF EST.IDEN = 'B'
THEN NEWEVT();
ELSE CHEEDA();
END;
ENDBLOCK;
ENDPROC;
% BUILDING THE REST OF THE TREE %

EXT PROC () Build:

ENT PROC Builder();
    WHILE EAST.IDEN # 'M' DO
        IF EAST.NEXTG == NILGA OR
            EAST.NEXTG == QM
            THEN IF EAST.NEXT == NILEV
            THEN EAST := EAST.BACKG.BACKG;
                GST := EAST.BACKG;
                BCST := GST.BC;
                UST := EAST.UNNO;
                GOTO L1;
            ELSE EAST := EAST.NEXT;
                GST := EAST.BACKG;
                UST := EAST.UNNO;
            END;
        END;
        FLAG := 1;
        C := 'f';
        Build();
    END;
L1: REP
    ENDPROC;

% BUILDING THE TREE %

EXT PROC () TOEV, BUILD, BUILDR, COPYGA;

ENT PROC BUILDT();
    IF BT = Ø
        THEN TOEV();
        BUILD();
    ELSE COPYGA();
        BUILDR();
        BUILD();
    END;
ENDPROC;
% BUILT THE TREE %

EXT PROC () SCANVA, DESIGN;

ENT PROC BUILTT(INT I);
RTD := 1;
IF RTD = 1 % REAL TIME ANALYSIS %
THEN SCANVA();
ELSEIF RTD = 0 % DESIGN ANALYSIS %
THEN DESIGN();
ELSE TWRT("\n\n*** ERROR IN BUILTT.RTL");
END:
ENDFROC;

PAGE 399
OPTION(1) LC,TR;
TITLE
CLEAR AND/OR LINK ARRAYS;

LET COMERR = 5000;
LET SP = OCT 42;
LET LF = OCT 12;

EXT PROC(INT) CLAOLI;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE ALMIP, COMIP;
EXT PROC (BYTE) ALMOP;
EXT PROC () INT IREAD;

SVC DATA RRERR:
  LABEL ERL;
  INT ERN;
  PROC(INT) ERP;
ENDDATA;

SVC DATA RRSIO:
  PROC () BYTE IN;
  PROC (BYTE) OUT;
ENDDATA;

SVC DATA RRERRX:
  INT LINENO;
  BYTE UEFLAG, ERLUN;
  INT RSXDSW;
ENDDATA;

DATA FILDAT:
  ARRAY (35) BYTE IFILE := "#SP(35)#";
  ARRAY (35) BYTE OFILE := "#SP(35)#";
ENDDATA;

ENT PROC RRJ0B();
  INT I;
  ERLUN := 3;
  IN := COMIP;
  IF IN( ) # 'C' OR IN( ) # 'A' OR IN( ) # 'L' OR IN( ) # ' ' 
    THEN ERP(COMERR);
  END;
  TREAD(OFILE,"=");
  TREAD(IFILE,"*" );
  I := IREAD();
  IN := ALMIP;
  OUT := ALMOP;
  CLAOLI(I);
ENDPROC;
% CHECK 'AND' GATE %

EXT PROC () REF VARIAB CHEVAR;
EXT PROC () GARCOL;
EXT PROC (BYTE, INT) INT GETBIB;

EXT PROC CHEAND() REF VARIAB;
WHILE GETBIB(GST-CLASS,1) = 1 DO
  GARCOL();
  REP;
  IF GST := NILGA THEN RETURN(NILVAN); END;
  IF EST.IDEN = 'M' OR EST.IDEN = 'SP'
  THEN RETURN(NILVAN);
END;
RETURN(CHEVAR());
ENDPROC;

PAGE CHEAND-RTL
% CHECK B, C, AND N, A, C, IN TREE

EXT PROC (INT, INT) INT GETBIT;
EXT PROC () REF VARIAB CHEAND, CHEVAR;

ENT PROC CHEBC(REF VARIAB V) REF VARIAB;
    IF EAST.IDEN = 'M' OR V :=: NILVAN
        THEN RETURN (V);
    END;

    BLOCK
        REF EVENT E :=: NILEV;
        REF GATE G :=: GST;
        REF BOUNDECO BCA :=: GST.BC;
        INT N, ANS;
        IF EST.IDEN # 'R' AND EST.IDEN # 'B'
            THEN WHILE G :=: NILGA DO
                E :=: G.BC.VARBC;
                IF V.NAMEF = E.NAVA.NAMEF AND
                    V.NAMET = E.NAVA.NAMET AND
                    V.NOVA = E.NAVA.NOVA
                    THEN GOTO LI;
                END;
                G :=: E.BACKG;
            END;
            G :=: E BACKG;
            RETURN (V);
        END;

    LI:N :=: EST.FAULT;
    IF N = E.FAULT
        THEN TURT("#NL#EVENT ALREADY IN THE SAME BRANCH#CR#");
    END;
    IF N <=: 16 THEN ANS :=: GETBIT(BCA, N);
    ELSEIF N <=: 32 THEN ANS :=: GETBIT(BCA, N - 16);
    ELSEIF N <=: 48 THEN ANS :=: GETBIT(BCA, N - 32);
    END;
    IF ANS = 1
        THEN IF GETBIT( GST.CLASS, 1) = 1
            THEN RETURN (CHEAND( ));
            ELSE EST :=: EST.NEXT;
                IF EST.IDEN = 'M' OR
                    EST.IDEN = SF
                    THEN RETURN (NILVAN)
                END;
        RETURN (CHEVAR( ));
    END;
    RETURN( V);
END;
% CHECK DIAMOND EVENT %

EXT PROC (REF EVENT, REF GATE) REF GATE DELGAT;

EXT PROC CHECKD();
    IF GST.NEXT := NILEV
    THEN GST := DELGAT (EAST, GST);
    BGST := GST.BC;
    EAST.NEXTG := NILGA;
    ELSE EAST.NEXT := NILEV;
    END;
ENDPROC;

% CHECK IF THE EVENT HAS BEEN DEVELOPED ALREADY %

EXT PROC () NEWEVENT;
EXT PROC (REF EVENT) NODEVUL;
EXT PROC (REF EVENT) INT CHSEBC;

INT PROC CHEEDA ();
    INT I := CI;
    REF EVENT E := EVT(I);
    WHILE E.IDEN <> SP DO
        IF E.NAVA <> NILVAR
            THEN BLOCK
                REF VARIAB V := E.NAVA;
                IF VST.NAMEF = V.NAMEF AND
                    VST.NAMET = V.NAMET AND
                    VST.NOVA = V.NOVA AND
                    EST.FAULT = E.FAULT
                    THEN IF CHSEBC (E) = 1
                        THEN NODEVUL(E);
                        ELSE NEWEVENT();
                    ELSE RETURN;
                END;
            END;
            RETURN;
        END;
    END;
ENDBLOCK;

I := I+1;
E := EVT(I);
REPE;
NEWEVENT();
ENDPROC;
% CHECK 'EX-OR' GATE %

EXT PROC (BYTE,INT) INT GETBIBJ
EXT PROC (REF VARIAB) REF VARIAB CHEEOR

ENT PROC CHEEOR (REF VARIAB V) REF VARIAB:
IF GETBIBJ(GST*CLASS,2) = 1
   THEN IF GST*NEXE #: NILEV
   THEN BLOCK:
      REF EVENT E := GST*NEXE;
      WHILE E*UNNO #: NILUNI DO
         IF E*NAVAMESU = 'Y'
            THEN
               TWRT("#NL#VARIABLE ALREADY ON"
               "(EX-OR GATE)";)
               OUT(E*NAVAMEF);
               OUT(E*NAVAMET);
               IWRT(E*NAVANOVA);
               TWRT("#NL#VALUE: ");
               IWRT(E*NAVAVALEVS.ACTVA);
               RETURN (NILVAN);
            END;
            TWRT("#NL#'VALUE: ");
            IWRT(E*NAVAVALEVS.ACTVA);
            RETURN (NILVAN);
         END;
         E := E*NEXT;
      END;
      RETURN (CHEEOR(V));
   END;
ENDPROC;
% CHECK ON-LINE VARIABLES %

EXT PROC (RAB) INT REFATR;

ENT PROC CHEONL(REF VARIAB V) INT;
    IF V.MESU = 'Y'
        THEN BLOCK
            REF DATAL DA := V.VALUES;
            IF DA.ACTVA > DA.HL
                THEN RETURN (REFATR("HI");
            ELSEIF DA.ACTVA < DA.LL
                THEN RETURN (REFATR("LO");
            END;
            RETURN (MAXNOFAULT + 1);
        ENDBLOCK;
    END;
    RETURN (0);
ENDPROC;
% CHECK IF VARIABLE IS ACTIVE %

ENT PROC CHEVAC(REF VARIAB V) INT:
    REF DATAV VI := V.VALUES;
    IF VI.ACTUA > VI.HL OR VI.ACTUA < VI.LL
    THEN RETURN (1);
    END;
    RETURN (0);
ENDPROC;
% CHECK VARIABLES %

EXT PROC (REF VARIAB) INT CHEONL;
EXT PROC (REF BYTE) REF VARIAB FIVIS;
EXT PROC (INT, REF VARIAB) REF VARIAB CHEVOR;
EXT PROC (REF VARIAB) REF VARIAB CHEBC;

INT PROC CHEVAR () REF VARIAB;
    INT F = 0;
    REF VARIAB V := NILVAR;
    LI: IF EST.IDEN # 'R' AND EST.IDEN # 'B'
        THEN V := FIVIS(EST.INOU);
            IF V := NILVAR
                THEN RETURN (V);
            ELSEIF V := NILVAR
                THEN RETURN (CHEBC(V));
            ELSEIF RTD = 1
                THEN F := CHEONL(V);
                    IF F # 0
                        THEN V := CHEVOR(F, V);
                        IF V := NILVAR
                            THEN IF EST.IDEN = 'M' OR
                                EST.IDEN = SP
                                THEN RETURN (NILVAR);
                            ELSEIF EST.NEXT.IDEN = 'M'
                                OR
                                EST.NEXT.IDEN = SP
                                THEN
                                    RETURN (NILVAR);
                            ELSE EST := EST.NEXT;
                                V := NILVAR;
                                GOTO LI;
                        ELSE RETURN (V);
                    END:
                    END;
            ELSE RETURN (V);
    END:
    END;
    RETURN (CHEBC(V));
ENDPROC;
% CHECK VARIABLES BEFORE PRINTING %

EXIT PROC(REF VARIA#) PRINVA;
EXIT PROC(BYTE, INT) INT GETBIB;

ENT PROC CHEVBP (REF STREAM S);
  REF VARIA# V := NILVAR;
  REF STREAM SA := NILSTM;
  IF GETBIB(S.SIO, 1) = 1
    THEN SA := S.FROMU:OUTST;
    WHILE SA !== NILSTM DO
      IF SA.TOUN !== S.TOUN
        THEN V := SA.NAVAR;
          GOTO L1;
      ELSE SA := SA.NEXTS;
      END;
    END;
  ELSE V := S.NAVAR;
  END;
END;

L1:TWRT("NL#VARIABLES : ");
  PRINVA(V);
  IF GETBIB(S.SIO, 1) = 1
    THEN IF S.NAVAR !== NILVAR
        THEN V := S.NAVAR;
          TWRT("NL#INTERNAL VARIABLES : ");
          PRINVA(V);
        END;
    END;
END;
ENDPROC;

PAGE
-------
CHEVBP.RTL
% CHECK VARIABLE OUT OF RANGE %

EXT PROC () REF VARIAB CHEAND;
EXT PROC (BYTE, INT) INT GETBIB;
EXT PROC (REF VARIAB) REF VARIAB CHEEOR;

EXT PROC CHEVOR(INT F, REF VARIAB V) REF VARIAB;
   IF F = MAXNOFAULT + 1 OR F ≠ ESTFAULT
      THEN IF GETBIB(GSTCLASS, 1) = 1
         THEN RETURN(CHEAN(D()));
         ELSE RETURN(NILVAN);
      END;
   END;
   RETURN(CHEEOR(V));
ENDPROC;
% CHECK IF B2, C2 ARE THE SAME %

EXT PROC (BYTE, INT) INT GETBI;
EXT PROC (REF BYTE, INT) CLBITB, SBITB;

EXT PROC CHSEBC(REF EVENT E) INT;
  REF BOUNDENO B1 := E.BALLG.BC, B2 := BCST;
  REF GATE G1 := E.BACKG, G2 := GST;
  REF EVENT E1 := G1.BACTG, E2 := G2.BACKE;
  INT FLAG1 := 0;
  IF B1 != NILBC
    THEN TWRNT("##IL##NILBC IN CHSEBC.RTL");
    RETURN(0);
  END;
  IF B1.NAI != B2.NAI
    THEN RETURN(0);
  ELSEIF B1.NA2 != B2.NA2
    THEN RETURN(0);
  ELSEIF B1.NA3 != B2.NA3
    THEN RETURN(0);
  END;
  WHILE EI.IDEN != 'M' DO
    WHILE E2.IDEN != 'M' DO
      IF GETBI(E2.INOU := 0
        THEN IF E1.VAR = E2.VAR AND
           E1.NAVAOVA = E2.NAVA.NOVA AND
           E1.FAULT = E2.FAULT
          THEN SBITB(E2.INOU, 7);
          GOTO L1;
        END;
      END;
      G2 := E2.BACKG;
      E2 := G2.BACKE;
      REP;
    L1:     G1 := E1.BACKG;
            E1 := G1.BACKE;
            G2 := GST;
            E2 := G2.BACKE;
            REP;
      L2: E2 := GST.BACKE;
      WHILE E2.IDEN != 'M' DO
        IF GETBI(E2.INOU := 0
          THEN FLAG1 := 1;
        END;
        CLBITB(E2.INOU, 7);
        G2 := E2.BACKG;
        E2 := G2.BACKE;
        REP;
      IF FLAG1 = 0 AND EI.IDEN = 'M'
        THEN RETURN(1);
      END;
    RETURN(0);
  ENDPROC;
% CLEAR ARRAYS AND/OR LINK THEM %

% 0- LINK ARRAYS ONLY %
% 1-CLEAR AND LINK THE PART OF THE ARRAYS THAT WAS USED IN %
% "THE CONSTRUCTION OF THE TREE %
% 2-CLEAR AND LINK THE WHOLE ARRAYS %

EXT PROC (INT) CLEVBT, CLEGUT;

ENT PROC CLAOI (INT NUM);
  CLEGUT(NUM);
  CLEVBT(NUM);
ENDPROC;

PAGE  CLAOI.RTL
CLEAR A BIT IN A BYTE

INT PROC CLBITB(REF BYTE BYT, INT BITNUM);
    VAL BYT := BYTE ( BYT LAND ((255 SLL BITNUM)) LOR 
                   (255 SRL (9-BITNUM)));
ENDPROC;
ENT PROC CLEBCT (INT NUM, CON);
    REF BOUNDECO B := NILBC;
    IF NUM = 0
        THEN GOTO L1;
    END;
    FOR I := CON TO MAXNOGM DO
        B := BCO(I);
        B*VARBC := NILEV;
        B*NA1 := 0;
        B*NA2 := 0;
        B*NA3 := 0;
        B*NEXTBC := NILBC;
        REP;
        FREEBC := BCO(CON);
    END;
L1: FOR I := CON TO MAXNOGM DO
    BCO(I)*NEXTBC := BCO(I+1);
    REP;
ENDPROC;
% CLEAR GATES USED IN BUILDING TREE %
% INCLUDES CLEARING A BIT OF VARIABLES USED IN THE TREE %

EXT PROC (INT, INT) CLEBCT;
EXT PROC (REF BYTE, INT) CLBITB;

EXT PROC CLEGUT (INT NUM);
INT CON := 1;
REF GATE G := NILGA;
REF UNITDE N := GAT(1) * BACKE * UNNO;
IF NUM = 0
THEN GOTO L1;
ELSEIF NUM = 1 THEN
    WHILE N :=: NILUNI DO
        CON := CON + 1;
        N := GAT(CON) * BACKE * UNNO;
    END;
END;
FOR I := CON TO MAXNOGMI DO
    G := GAT(I);
    G * CLASS := 2;
    G * EXTRA := SP;
    G * BACKE := NILEV;
    G * NEXTE := NILEV;
    G * NEGATE := NILGA;
    G * BC := NILBC;
END;
FOR I := 1 TO MAXNOVAR DO
    CLEBCT(VAB(I) * EX, 2);
END;
FREEGA := GAT(CON);
GATEPO := GAT(CON);
L1: FOR I := CON TO MAXNOGMI DO
    GAT(I) * NEGATE := GAT(I + 1);
END;
CLEBCT(NUM, CON);
ENDPROC;
ENT PROC CLEVBT(INT NUM);
  INT CON := 1;
  REF EVENT E := NILEV;
  IF NUM = 0 THEN GOTO L1;
  ELSEIF NUM = 1 THEN CON := C1;
END;
FOR I := CON TO MAXNOEM1 DO
  E := EVT(I);
  E.IDEN := SP;
  E.UNID := SP;
  E.VAR := SP;
  E.INOU := 0;
  E.NAVA := NILVAR;
  EFAULT := 0;
  E.NEXT := NILEV;
  E.TOP := NILEV;
  E.BACKG := NILGAI;
  E.NEXTG := NILGAI;
  E.UNNO := NILUNI;
  E.PROB := 0.0B0;
  REP;
  FREEV := EVT(CON);
  BT := 0;
  C := '+';
  FLAG := 0;
L1: FOR I := CON TO MAXNOEM1 DO
    EVT(I).NEXT := EVT(I+1);
  REP;
ENDPROC;
CLEAR A BIT IN AN INTEGER %

ENT PROC CLRBIT(REF INT WORD, INT BITNUM);
  VAL WORD := WORD LAND ((-1 SLL BITNUM) LOR
                       (-1 SRL(17-BITNUM)));
ENDPROC;
% COMMAND INPUT %

LET BUflen = 80;
LET GMCERR = 5043;

EXT PROC (REF ARRAY BYTE) RSGMC;
EXT PROC (INT) TESDsw;

DATA COMDAT;
   ARRAY(BUflen)BYTE BUF := (SP(BUflen));
   INT INEXT := 0;
   TOTAL := 0;
ENDDATA;

ENT PROC COMIP()BYTE;
   INEXT := INEXT + 1;
   IF INEXT = 1
      THEN RSGMC(BUF);
      TESDsw(GMCERR);
      TOTAL := RSGDSW;
   END;
   IF INEXT <= TOTAL
      THEN RETURN(BUF(INEXT));
   END;
RETURN(LF);
ENDPROC;

PAGE

-------

COMIP *RTL
EXIT PROC(REF BYTE, INT, RAB) GETBYTE;

ENT PROC COMPVA(REF STREAM AS, REF UNITDE U);
  WHILE AS !== NILSTM DO
    IF AS.TOUN === U
      THEN
        BLOCK
          REF VARIAB V := AS.NAVAR;
          WHILE V !== NILVAR DO
            GETBYTE(V.NAMET, 1, "NAME OF VARIABLE");
            V := V.NEXVAR;
          END;
        ENDBLOCK;
        END;
        AS := AS.NEXTS;
      END;
    END;
  END PROC;
% COMPLETE VARIABLES FOR UNITS NOT BUILT YET %

EXT PROC(REF BYTE, INT, RAB) GETBYT;

EXT PROC () REF VARIAB NEXTFV;

EXT PROC COMPV1(REF STREAM S);

BYTE ANS;
   REF VARIAB V := NILVAR, V1 := S.NAVAR;
   L1: GETBYT(ANS, 2, "ARE THERE ANY MORE VARIABLES(Y/N)");
       WHILE ANS = 'Y' DO
           V := NEXTFV();
           GETBYT(V.NAMET, 1, "NAME OF VARIABLE");
           V.NEXVAR := NILVAR;
           V1.NEXVAR := V;
           V1 := V;
           GETBYT(ANS, 2, "ARE THERE ANY MORE VARIABLES(Y/N)");
           REP;
           IF ANS # 'N'
               THEN TWRT("\nPLEASE TYPE(Y/N)");
           END:
       END:
   ENDPROC;

PAGE

COMPV1.RTL
ENT PROC COMRAB(RAB A, B) INT;
    IF LENGTH B > LENGTH A THEN GOTO LI; END;
    FOR I := 1 TO LENGTH B DO
        IF A(I) ≠ B(I) THEN RETURN(0); END;
    REP;
    RETURN(1);
LI: Twrt("\nSTRING TOO LONG");
    RETURN(0);
ENDPROC;
% COPY BITS OF A BYTE %

INT PROC COPBI B (BYTE B1, REF BYTE B2);
VAL B2 := B2 LOR B1;
ENDPROC;
% COPY BOUND, COND, AND N· A· C· %

EXT PROC () REF BOUNDECO NEXTFB;
EXT PROC (INT, REF INT) COPYBI;

EXT PROC COPYBC(REF GATE G);
    REF BOUNDECO C1 := EST·NEXTG·BC;
    REF BOUNDECO C2 := NEXTFB();
    C2·VARBC := EAST;
    COPYBI(C1·NA1·C2·NA1);
    COPYBI(C1·NA2·C2·NA2);
    COPYBI(C1·NA3·C2·NA3);
    COPYBI(BCST·NA1·C2·NA1);
    COPYBI(BCST·NA2·C2·NA2);
    COPYBI(BCST·NA3·C2·NA3);
    G·BC := C2;
    BCST := C2;
ENDPROC;
| COPY BITS OF AN INTEGER |

INT PROC COPYBI(INT C1, REF INT C2);
  VAL C2 := C2 LOR C1;
ENDPROC;
% COPY A GATE %

EXT PROC () REF GATE NEXTFG;
EXT PROC (BYTE, REF BYTE) COPBIB;
EXT PROC (REF GATE) COPYBC;

ENT PROC COPYGA();
   REF GATE G := NEXTFG();
   COPBIB(EAST,NEXTG,CLASS,G,CLASS);
   G.BACKE := "EAST";
   EAST.NEXTG := G;
   COPYBC(G);
   GST := G;
ENDPROC;

--------
% GET THE DATA OF THE VARIABLE

EXT PROC(BYTE, INT) INT GETBIB;
EXT PROC(REF BYTE, INT) SBITBY;
EXT PROC() REF VARIAB NEXTFV;
EXT PROC(REF BYTE, INT, RAB) GETBYT;
EXT PROC(REF INT, INT, INT, RAB, LABEL) GETINT;
EXT PROC(REF VARIAB) VALUE;

EXT PROC DATVAR(REF UNITDE U, REF STREAM S);
REF VARIAB V := NILVAR;
IF S::VAR := NILVAR
  THEN S::VAR := NEXTFV();
  SBITBY(S::EO, 1);
END;
V := S::VAR;
WHILE GETBIB(V::EX, 1) ≠ Ø DO
  IF V::NEXVAR := NILVAR
    THEN V::NEXVAR := NEXTFV();
    V := V::NEXVAR;
  ELSE V := V::NEXVAR;
END;

REP;
GETBYT(V::NAMEF, 1, "NAME OF VARIABLE");
L1: GETINT(V::NOVA, 1, "NO. OF THE VARIABLE", L1);
  V::BACKS := S;
L2: GETBYT(V::MESU, 1, "IS THE VARIABLE MEASURED (Y/N)");
  IF V::MESU = 'Y'
    THEN VALUE(V);
  ELSEIF V::MESU ≠ 'N'
    THEN TWRTC("PLEASE TYPE (Y/N)");
    GOTO L2;
END;
SBITBY(V::EX, 1);
IF V::NEXVAR::NAME = SF AND V::NEXVAR::NOVA = Ø
  THEN V::NEXVAR := NILVAR;
END;
ENDPROC;

PAGE DATVAR.RTL
OPTION(1) BC, TR;
TITLE
DEBUG;

LET COMERR = 5280;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC(INT, INT) DEBUG;
EXT PROC() RSXINT);
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOPI, RSXOP0;
EXT PROC(INT) INT RSXSWI, RSXSW0;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE COMIP;
EXT PROC () INT 1READ;

SVC DATA RRERR:
   LABEL ERL;
   INT ERN;
   PROC(INT) ERF;
ENDDATA;

SVC DATA RRSS0;
   PROC () BYTE IN;
   PROC (BYTE) OUT;
ENDDATA;

SVC DATA RRERRX:
   INT LINENO;
   BYTE UEFLAG, ERRLUN;
   INT RSXDSW;
ENDDATA;

DATA FILEDAT:
   ARRAY (35) BYTE IFILE := "#SP(35)#";
   ARRAY (35) BYTE OFILE := "#SP(35)#";
ENDDATA;

ENT PROC RRJOB();
   INT I, J;
   % I- NO. OF ARRAY TO PRINT %
   % J- PART OF THE ARRAY TO PRINT %
   ERRLUN := 3;
   IN := COMIP;
   IF INC() # 'D' OR INC() # 'E' OR INC() # 'B' OR INC() # '
    THEN ERP(COMERR);
END;

TREAD(OFILE, ";");
TREAD(IFILE, ";");
I := IREAD();
J := IREAD();
RSXINT();
RSXOPI(1, OFILE);
RSXSW0(1);
RSXOPI(2, IFILE);
RSXSWI(2);
DEBUG(I, J);
RSXCL();
RSXCL (2); ENDPROC;
% DEBUG ARRAY OF BOUNDARY CONDITIONS %

EXT PROC(Int,Int,Byte) PRICHAI;
EXT PROC(Int) DEBBCP;

ENT PROC DEBBC (Int J);
    INT I := 1;
    TWRT("\#NL(3)#ARRAY OF BOUNDARY CONDITIONS");
    PRICHAI(1,21,'*');
    SWITCH J OF L1,L2,L3;
    TWRT("\#NL#ILLEGAL VALUE");
    GOTO EX;
L1: WHILE I <= MAXNOG DO
   DEBBCP(I);
   I := I+1;
   REP;
   GOTO EX;
L2: BLOCK
   REF UNITDE UBC := BCO(1)*VARBC.UNNO;
   INT K := 1;
   WHILE UBC := NILUNI DO;
      DEBBCF(K);
      K := K+1;
      UBC := BCO(K)*VARBC.UNNO;
   REP;
   ENDBLOCK;
   GOTO EX;
L3: BLOCK
   REF UNITDE UBC := BCO(1)*VARBC.UNNO;
   INT L := 1;
   WHILE UBC := NILUNI DO
      UBC := BCO(L)*VARBC.UNNO;
      L := L+1;
   REP;
   WHILE UBC == NILUNI DO
      DEBBCP(L);
      L := L+1;
      UBC := BCO(L)*VARBC.UNNO;
   REP;
   ENDBLOCK;
EX: RETURN;
ENDPROC;
% DEBUG & PRINT ARRAY OF BOUND. COND. %

EXT PROC (RAb&, INT, REF BYTE) TESTSP;
EXT PROC (INT, INT) PRIFFB;

ENT PROC DEBbCP (INT I);
    REF BOUNDECO BC := BCO(I);
    TWRT("#NL(3)#B. C. : ");
    IWRT(I);
    TWRT("#NL#VARBC : ");
    IF BC#VARBC := 1 NILEV
      THEN TWRT("NIL EVENT");
    ELSE
      FOR K := 1 TO MAXNOE DO
        IF BC#VARBC :=: EVT(K)
          THEN IWRT(K);
          GOTO L1;
      END;
      REP;
      END;
      L1: TWRT("#NL#NA1 : ");
      PRIFFB(BC#NA1, 1);
      TWRT("#NL#NA2 : ");
      PRIFFB(BC#NA2, 2);
      TWRT("#NL#NA3 : ");
      PRIFFB(BC#NA3, 3);
      TWRT("#NL#NEXTBC: ");
      IF BC#NEXTBC :=: NILNEBC
        THEN TWRT("NIL NEXT BC");
      ELSEIF BC#NEXTBC :=: NILBC
        THEN TWRT("NIL BC");
      ELSE
        FOR J := 1 TO MAXNOG DO
          IF BC#NEXTBC :=:= BCO(J)
            THEN IWRT(J);
          END;
        REP;
      END;
      END;
      ENDPROC;
% DEBUG & PRINT ARRAY OF DATA FOR VARIABLES %

EXT PROC RAB, INT REF BYTE) TESTSP;
EXT PROC (INT INT REF BYTE) PRICHA;

EXT PROC DEBDAT ();
TWRT("#NL(3)#ARRAY OF DATA FOR VARIABLES");
PRICHA(1, 27, '*');
FOR I := 1 TO MAXNOMEVAR DO
  TWRT("#NL(3)#DAT: ");
  IWRT(I);
  BLOCK
    REF DATAV D := DAT(I);
    INT K;
    TWRT("#NL#HL: ");
    IWRT(D#HL);
    TWRT("#NL#LL: ");
    IWRT(D#LL);
    TWRT("#NL#PRVA: ");
    IWRT(D#PRVA);
    TWRT("#NL#ACTVA: ");
    IWRT(D#ACTVA);
    TWRT("#NL#P10: ");
    K := D#P10;
    IWRT(K);
    TESTSP("PREX: ", I, D#PREX); END;
  "END_BLOCK;
  REF;
ENDPROC;
% DEB:UG ARRAY OF EVENTS %

EXT PROC (INT, INT, BYTE) PRICH A;
EXT PROC (INT) DEBEVP;

EXT PROC DEBEV (INT J);
TWRT("\#NL(3)#ARRAY OF EVENTS");
PRICH A (1, 15, '*');
SWITCH J OF L1, L2, L3;
TWRT("\#NL#ILLEGAL VALUE");
GOTO EX;
% 1=PRINT THE WHOLE ARRAY %
% 2=PRINT THE EVENTS RELATED TO THE MINITREES %
% 3=PRINT THE EVENTS RELATED TO THE TREE %

L1: FOR I := 1 TO MAXNOE DO
    DEBEV (I);
    REP;
    GOTO EX;
L2: BLOCK
    REF UNITDE U := EVT(I).UNNO;
    INT K := I;
    WHILE U != NILUNI DO
        DEBEVP(K);
        K := K + 1;
        U := EVT(K).UNNO;
    REP;
    ENDBLOCK;
    GOTO EX;
L3: BLOCK
    REF UNITDE U := EVT(I).UNNO;
    INT L := 1;
    WHILE U != NILUNI DO
        L := L + 1;
        U := EVT(L).UNNO;
    REP;
    WHILE U != NILUNI DO
        DEBEVP(L);
        L := L + 1;
        U := EVT(L).UNNO;
    REP;
    ENDBLOCK;
    EX: RETURN;
ENDPROC;
% DEBUG ARRAY OF EVENTS AND PRINT %

EXT PROC (FRAC, INT) FWRTF;
EXT PROC (BYTE, INT) INT GETBIB;
EXT PROC (RAE, INT, REF BYTE) TESTSP;
EXT PROC (INT) TRNOWR;

EXT PROC DEBEVP (INT I);
INT KU, KI;
REF EVENT E := EVT(I);
TWRT("#NL(3)#EVENT : ");
IWR'T(I);
TESTSP("IDENT: "; E.IDENT);
TWRT("#NL#UNID : ");
IF E.UNID = SP
  THEN OUT('S');
ELSE KU := E.UNID;
  IWR'T(KU);
  TWRT("#SP(3)#");
  TWRT(TYUCKU); L;
END;
TESTSP("VAR: "; E.VAR);
TWRT("#NL#INOU: ");
FOR I := 1 TO 8 DO
  IF GETBIB(E.INOU, I) = 1
    THEN IWR'T(I);
  ELSE OUT('0');
END;
REP;
TWRT("#NL#NAV: ");
IF E.NAVA = NILVAR
  THEN TWRT("NIL VARIABLE");
ELSE IWR'T(E.NAVA.NAVA);
END;
TWRT("#NL#FAULT: ");
TRNOWR(E.FAULT);
TWRT("#NL#NEXT: ");
IF E.NEXT = NILNEX
  THEN TWRT("NIL NEXT EVENT");
ELSEIF E.NEXT := NILEV
  THEN TWRT("NIL EVENT");
ELSE FOR K := 1 TO MAXNOE DO
  IF E.NEXT := EVT(K)
    THEN IWR'T(K);
    GOTO L1;
END;
REP;
L1: TWRT("#NL#TOP: ");
  IF E.TOP := NILEV
    THEN TWRT("NIL EVENT");
  ELSE FOR L := 1 TO MAXNOE DO
    IF E.TOP := EVT(L)
      THEN IWR'T(L);
      GOTO L2;
END;
REP;
L2: TWRT("#NL#BACKG: ");
  IF E.BACKG := NILGA
    THEN TWRT("NIL GATE");

PAGE DEBEVP.RTL
ELSE FOR $M := 1$ TO MAXNOG DO
  IF $E\cdot$BACKG := $GAT(M)$
  THEN $IWR(M)$;
  GOTO L3;
END;
REP;
END;
L3: $TWRT("\#NL\#NEXTG: ");$
  IF $E\cdot$NEXTG := $NILGA$
  THEN $TWRT("NIL GATE");$
  ELSEIF $E\cdot$NEXTG := $QM$
  THEN $TWRT("QM");$
  ELSE FOR $N := 1$ TO MAXNOG DO
    IF $E\cdot$NEXTG := $GAT(N)$
    THEN $IWR(N)$;
    GOTO L4;
  END;
  REP;
END;
L4: $TWRT("\#NL\#UNNO: ");$
  IF $E\cdot$UNNO := $NILUNI$
  THEN $TWRT("NIL UNIT");$
  ELSE FOR $F := 1$ TO MAXNOUNIT DO
    IF $E\cdot$UNNO := $UNI(F)$
    THEN $IWR(F)$;
    GOTO L5;
  END;
  REP;
END;
L5: $TWRT("\#NL\#PROB: ");$
  $FWRT(E\cdot$PROB, 4)$;
ENDPROC;
EXIT PROC(int, int, byte) PRICHAS;

EXIT PROC DEBFAU();
TWRT("#NL(3)#ARRAY OF FAULTS");
PRICHAS(1, 15, '*');
FOR I := 1 TO MAXNOFAULT DO
  TWRT("#NL(2)#FAULT (");
  IWRT(I);
  TWRT('"');
  TWRT(FAU(I) + L);
REP;
ENDPROC;
% DEBUG ARRAY OF GATES %

EXT PROC(INT, INT, BYTE) PRICH;
EXT PROC(INT) DEBGAP;

ENT PROC DEBGA ( INT J):
TWRT("#NL(3)#ARRAY OF GATES");
PRICH(1, 14, '*');
SWITCH J OF L1, L2, L3:
TWRT("#NL#ILLEGAL VALUE");
GOTO EX;
L1: FOR I := 1 TO MAXNOG DO
   DEBGAP(I);
   REP;
GOTO EX;
L2: BLOCK
   REF UNITDE UG := GAT(1)*BACK*UNNO;
   INT K := 1;
   WHILE UG /= NILUNI DO:
      DEBGAP(K);
      K := K+1;
      UG := GAT(K)*BACK*UNNO;
   REP;
ENDBLOCK;
GOTO EX;
L3: BLOCK
   REF UNITDE UG := GAT(1)*BACK* UNNO;
   INT L := 1;
   WHILE UG /= NILUNI DO
      L := L+1;
      UG := GAT(L)*BACK*UNNO;
   REP;
   WHILE UG /= NILUNI DO
      DEBGAP(L);
      L := L+1;
      UG := GAT(L)*BACK*UNNO;
   REP;
ENDBLOCK;
EX: RETURN;
ENDPROC;
% DEBUG & PRINT ARRAY OF GATES %

EXT PROC (RAB, INT, REF BYTE) TESTSP;
EXT PROC (BYTE, INT) INT GETBIB;

ENT PROC DEBUG (INT I);
  INT bN;
  REF GATE Ga := GAT (I);
  TWRT("#NL(3)#GATE = ");
  IWRT (I);
  TWRT("#NL#CLASS = ");
  IF Ga BACK := NILEV
    THEN IF Ga CLASS = 0 THEN OUT ('Z') END;
    ELSE IWRT (GETBIB (Ga CLASS, 2));
    TWRT("#SP(3)#");
    IWRT (GETBIB (Ga CLASS, 1));
  END;
  TESTSP("EXTRA: ", Ga EXTRA);
  TWRT("#NL#BACK: ");
  IF Ga BACK := NILEV
    THEN TWRT("NIL EVENT");
    ELSE FOR J := 1 TO MAXNOE DO
      IF Ga BACK := EVT(J)
        THEN IWRT (J);
        GOTO L1;
      END;
      END;
    REP;
  END;

L1: TWRT("#NL#NEXTE = ");
  IF Ga NEXTE := NILEV
    THEN TWRT("NIL EVENT");
    ELSE FOR K := 1 TO MAXNOE DO
      IF Ga NEXTE := EVT(K)
        THEN IWRT (K);
        GOTO L2;
      END;
      END;
    REP;
  END;

L2: TWRT("#NL#NEGATE = ");
  IF Ga NEGATE := NILNEG
    THEN TWRT("NIL NEXT GATE");
    ELSE IF Ga NEGATE := NILGA
      THEN TWRT("NIL GATE");
      ELSE FOR L := 1 TO MAXNOE L3
        IF Ga NEGATE := GAT (L)
          THEN IWRT (L);
          GOTO L3;
        END;
        END;
      REP;
    END;

L3: TWRT("#NL#BC = ");
  IF Ga BC := NILEC
    THEN TWRT("NIL BOUND. COND.");
    ELSE FOR J := 1 TO MAXNOG DO
      IF Ga BC := BCO (J)
        THEN IWRT (J);
        GOTO L4;
      END;
      END;
    REP;
  END;
L4: RETURN;
ENDPROC;
% DEBUG & PRINT ARRAY OF INDEX TO LOCATE TOP EVENTS %

EXT PROC (RAJ, INT, REF, BYTE) TESTSP;
EXT PROC (INT, INT, BYTE) PRICHAI;

ENT PROC DEBLOC();
INT TA;
TWRT("#NL(3)#ARRAY TO LOCATE TOP EVENTS");
PRICHAI(1, 26, '*');
FOR I := 1 TO MAXNOUNIT DO
TWRT("#NL(3)#");
TWRT("LOT: ");
IWR(T(I));

BLOCK
REF INDEX LO := LOT(I);
TWRT("#NL#TYPE: ");
IF LO::TYPE = SP
THEN OUT('S');
ELSE TA := LO::TYPE;
TWRT(TYU(TA), L);
END;
TESTSP("TEX: " + LO::TEX);
TWRT("#NL#FIRSEL: ");
IF LO::FIRSEL == NILEV
THEN TWRT("NIL EVENT");
ELSE FOR J := 1 TO MAXNOE DO
IF LO::FIRSEL == EVT(J)
THEN IWR(T(J));
GOTO LI;
END;
ENDBLOCK;
END;
ENDPROC;

PAGE DEBLOCRTL

---------
% DEBUG & PRINT ARRAY OF STREAMS%

EXT PROC (INT, INT, BYTE) PRICHA;
EXT PROC (BYTE, INT) INT GETBIB;
EXT PROC (RAB, INT, REF BYTE) TESTSP;

ENT PROC DEBSTR();
  TWRT("#NL(3)#ARRAY OF STREAMS");
  PRICHA(1, 16, ";");
  FOR I := 1 TO MAXNOSTR DO
    TWRT("#NL(3)#");
    TWRT("STREAM: ");
    IWRT(I);
    BLOCK
      REF STREAM S := STR(I);
      TWRT("#NL#SIB: ");
      IF S.FROMU := NILUNI
        THEN OUT("Z");
      ELSE IWRT(GETBIB(S.510, 1));
    END;
    TWRT("#NL#EIO: ");
    IF S.FROMU := NILUNI
      THEN OUT("Z");
    ELSE IWRT(GETBIB(S.EIO, 2));
      TWRT("#SP(3)#");
      IWRT(GETBIB(S.EIO, 1));
    END;
    TWRT("#NL#FROMU: ");
    IF S.FROMU := NILUNI
      THEN TWRT("NIL UNIT");
    ELSE FOR J := 1 TO MAXNOUNIT DO
      IF S.FROMU := UNI(J)
        THEN IWRT(J);
        GOTO L1;
    END;
    REF;
  END;
L1: TWRT("#NL#TOUN: ");
  IF S.TOUN := NILUNI
    THEN TWRT("NIL UNIT");
  ELSE FOR K := 1 TO MAXNOUNIT DO
    IF S.TOUN := UNI(K)
      THEN IWRT(K);
      GOTO L2;
  END;
  REP;
END;
L2: TWRT("#NL#NAVAR: ");
  IF S.NAVAR := NILVAR
    THEN TWRT("NIL VARIABLE");
  ELSE FOR L := 1 TO MAXNOVAR DO
    IF S.NAVAR := VAB(L)
      THEN IWRT(L);
      GOTO L3;
  END;
  REP;
END;
L3: TWRT("#NL#NEXTS: ");
  IF S.NEXTS := NILSTM
    THEN TWRT("NIL NEXT STREAM");
  ELSEIF S.NEXTS := NILSTM
    THEN TWRT("NIL NEXT STREAM");
  ELSE FOR M := 1 TO MAXNOSTR DO
    IF S.NEXTS := STR(M)
      THEN IWRT(M);
      GOTO L1;
  END;
  REP;
END;
THEN TWRT("NIL STREAM");
ELSE FOR N := 1 TO MAXNOSTR DO
  IF S.NEXTS := STR(N)
  THEN LWRT(N);
  GOTO L4;
END;
REPI
END;
ENDEBLOC;
L4:REP;
ENDP:PROC;
EXT PROC (INT, INT, BYTE) PRICHA;

ENT PROC DEBTUN ();
  TWRT("#NL(3)#ARRAY OF TYPE OF UNITS");
  PRICHA(1, 23, ' '); 
  FOR I := 1 TO MAXNOUNIT DO 
    TWRT("#NL(2)#TYPE OF UNIT (");
    IWRT(I);
    TWRT("\n") ; ");
    TWRT(TYU(I).L);
  REP;
ENDPROC;
% PRINT THE DATA BASE %
EXT PROC(INT) DEBEV, DEBGA, DEBBCC;
EXT PROC() DEBUN, DEBSTR, DEBVAR, DEBLOC, DEBDAT, DEBFAU, DEBTUN;

ENT PROC DEBUG(INT I, INT J);
    SWITCH I OF L1, L2, L3, L4, L5, L6, L7, L8, L9, L10, L11;
    SWRT("#NL#ILLEGAL VALUE");
    GOTO Q1;
    % 1-ARRAY OF EVENTS %
    % 2-ARRAY OF GATES %
    % 3-ARRAY OF BOUND. COND. %
    % 4-ARRAY OF TOP EVENTS %
    % 5-ARRAY OF UNITS %
    % 6-ARRAY OF STREAMS %
    % 7-ARRAY OF VARIABLES %
    % 8-ARRAY OF DATA FOR MEASURED VARIABLES %
    % 9-ARRAY OF FAULTS %
    %10-ARRAY OF TYPES OF UNITS %
    %11-PRINT ALL THE ARRAYS %
L1: DEBEV(J);
    GOTO Q1;
L2: DEBGA(J);
    GOTO Q1;
L3: DEBBCC(J);
    GOTO Q1;
L4: DEBLOC();
    GOTO Q1;
L5: DEBUN();
    GOTO Q1;
L6: DEBSTR();
    GOTO Q1;
L7: DEBVAR();
    GOTO Q1;
L8: DEBDAT();
    GOTO Q1;
L9: DEBFAU();
    GOTO Q1;
L10: DEBTUN();
    GOTO Q1;
L11: DEBEV(J);
    DEBGA(J);
    DEBBCC(J);
    DEBLOC();
    DEBUN();
    DEBSTR();
    DEBVAR();
    DEBDAT();
    DEBFAU();
    DEBTUN();
Q1: RETURN;
ENDPROC;
% DEBUG & PRINT ARRAY OF UNITS %

EXT PROC (RAB, INT, REF BYTE) TESTSP;
EXT PROC (INT, INT, BYTE) PRICHA;

ENT PROC DEBUG();
    INT TA;
    TWRT("#NL(3)#ARRAY OF UNITS");
    PRICHA(1, 14, '*');
    FOR I := 1 TO MAXNOUNIT DO
        TWRT("#NL(3)#");
        TWRT("UNIT: ");
        IWRT(I);
    END;
    BLOCK
        REF UNITDE U := UNI(I);
        TWRT("#NL#UNUM: ");
        IWRT(U#UNUM);
        TWRT("#NL#TY: ");
        IF U#TY = SF
            THEN OUTC('S');
        ELSE TA := U#TY;
            TWRT(TYUC(TA)*L);
        END;
        TESTSP("TYE: ", U#TYE);
        TWRT("#NL#INST: ");
        IF U#INST := NILSTM
            THEN TWRT("NIL STREAM");
        ELSE FOR J := 1 TO MAXNOSTR DO
            IF U#INST# := STR(J)
                THEN IWRT(J);
            GOTO L1;
        END;
        REP;
    END;
    IF U#INST := NILINS
        THEN TWRT("NILINS");
    END;
L1: TWRT("#NL#OUTST: ");
    IF U#OUTST := NILSTM
        THEN TWRT("NIL STREAM");
    ELSE FOR K := 1 TO MAXNOSTR DO
        IF U#OUTST# := STR(K)
            THEN IWRT(K);
        GOTO L2;
    END;
    REP;
    END;
    IF U#OUTST := NILOUT
        THEN TWRT("NILOUT");
    END;
L2: TWRT("#NL#NEXU: ");
    IF U#NEXU := NILUNE
        THEN TWRT("NIL NEXT UNIT");
    ELSEIF U#NEXU := NILUNIT
        THEN TWRT("NIL UNIT");
    ELSE FOR M := 1 TO MAXNOUNIT DO
        IF U#NEXU# := UNI(M)
            THEN IWRT(M);
        GOTO L3;
    END;
    REP;
    END;

PAGE: DEBUG.RTL
% DEBUG & PRINT ARRAY OF VARIABLES %

EXT PROC (RAB, INT, REF BYTE) TESTSP;

EXT PROC (BYTE, INT) INT GETBIB;

EXT PROC (INT, INT, BYTE) PRICHA;

EXT PROC DEBVAR();
TWRT("@NL(3)#ARRAY OF VARIABLES");
PRICHA(1, 18, '*');
FOR I := 1 TO MAXNOVAR DO
TWRT("@NL(3)#");
TWRT("VAR: ");
IWRT(I);
END;

IWRT(V*NOVA);
TWRT("@NL@NOVA");
IWRT(V*NOVA);

TWRT("@NL@BACKS");
IF V*BACKS := NILSTM
THEN TWRT("NIL STREAM");
ELSE FOR J := 1 TO MAXNOSTR DO
IF V*BACKS := STR(J)
THEN IWRT(J);
GOTO L1;
END;
REP;

L1: TWRT("@NL@NEXTVAR");
IF V*NEXTVAR := NILVAR
THEN TWRT("NIL NEXT VAR");
ELSE IF V*NEXTVAR := NILVAR
THEN TWRT("NIL VARIABLE");
ELSE FOR K := 1 TO MAXNOVAR DO
IF V*NEXTVAR := VAB(K)
THEN IWRT(K);
GOTO L2;
END;
REP;

L2: TWRT("@NL#VALUES");
IF V*VALUES := NILDAT
THEN TWRT("NILDAT");
ELSE FOR L := 1 TO MAXNAMEVAR DO
IF V*VALUES := DAT(L)
THEN IWRT(L);
GOTO L3;
END;
REP;

END;
ENDBLOCK;
L3: REP;
ENDPROC;
% DELETE A B C %
ENT PROC DELEBC (REF GATE G);
    REF BOUNDECO B := G.BC;
    B.VARBC := NILEV;
    B.NA1 := Ø;
    B.NA2 := Ø;
    B.NA3 := Ø;
    B.NEXTBC := FREEBC;
    FREEBC := B;
ENDPROC;

% DELETE AN EVENT

ENT PROC DELEVT (REF EVENT A,REF GATE G) REF EVENT;
REF EVENT E := A,E1 := NILEV;
IF E:UNNO :#; NILUNI OR E:#: NILEV
THEN E:IDEN := SP;
   E:UNID := SP;
   ESVAR := SP;
   E:INOU := Ø;
   E:NAVAA := NILVAR;
   E:FAULT := Ø;
   E1 := E:NEXT;
   E:NEXT i := FREEV;
   FREEV := E;
   E:TOP := NILEV;
   E:BACKG := NILGA;
   E:NEXTG := NILGA;
   UST := E:UNNO;
   E:UNNO := NILUNI;
   E:PROB := Ø:888;
   E" := E1;
ELSE G:NEXT := NILEV;
   E" := G:BACK;
END;
RETURN (E);
ENDPROC;
% DELETE A GATE %

EXT PROC (REF GATE) DELBC;

ENT PROC DELGAT (REF EVENT E; REF GATE G) REF GATE:

REF GATE G1 := G;
G := E.BACk;
G1.CLASS := 0;
G1.EXTRA := SP;
G1.BACKE := NILEV;
G1.NEXTE := NILEV;
G1.NEGATE := FREEGA;
FREEGA := G1;
DELBC(G1); "
G1.BC := NILBC;
RETURN (G);
ENDPROC;

PAGE DELGAT.RTL
OPTION(1) BC,TR;
TITLE
DESCRIPTION OF THE SYSTEM;

LET COMERR = 5000;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC( ) DESUN;
EXT PROC( ) RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOF1, RSXOPO;
EXT PROC(INT) INT RSXSWI, RSXSWO;
EXT PROC( REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC( ) BYTE ECHOIN, COMIP;

SVC DATA RRERR;
  LABEL ERL;
  INT ERN;
  PROC(INT) ERP;
ENDDATA;

SVC DATA RRSIO;
  PROC( ) BYTE IN;
  PROC( BYTE) OUT;
ENDDATA;

SVC DATA RRERRX;
  INT LINENO;
  BYTE UEFLAG, ERRLUN;
  INT RSXDSW;
ENDDATA;

DATA FILDAT;
  ARRAY(78) BYTE FILE := "#SP(70)#";
ENDDATA;

EXT PROC RRJOB();
  ERRLUN := 3;
  RSXINT();
  IN := COMIP;
  IF IN() # 'D' OR IN() # 'E' OR IN() # 'S' OR IN() # ' ' THEN ERP(COMERR);
END;

TREAD(FILE,"=");
RSXOPO(1, FILE);
RSXSWO(1);

FOR J := 1 TO LENGTH FILE DO
  FILE(J) := SP;
REP;

TREAD(FILE,"#LF#" );
IN := ECHOIN;
RSXOPI(2, FILE);
RSXSWI(2);
DE. UN( );
RSXCL(1);
RSXCL(2);
% BUILDING THE TREE FOR DESIGN %

EXT PROC (REF INT, INT, INT, INT, RAB, LABEL) GETINT;
EXT PROC (REF BYTE, INT, RAB) GETBYTE;
EXT PROC (RAB, BYTE, INT, RAB, LABEL) GETSTR;
EXT PROC (RAB, RAB) INT COMRA;
EXT PROC (INT, INT, BYTE, BYTE) LOOKFV;

ENT PROC DESIGN();
    INT FAULTD, NOVAR;
    BYTE NAMF, NAMT;
    TWRT("#NL(5)#TREE FOR DESIGN");
L1: GETSTR(ST, '"', 2, "NAME OF VARIABLE (ADD \ at the end)", L1);
L2: GETINT(NOVAR, MAXNOVAR, "NO. OF THE VARIABLE", L2);
    NAMF := ST(1);
    NAMT := ST(2);
L3: GETSTR(ST, '"', 1, "FAULT", L3);
    FOR I := 1 TO MAXNOFAULT DO
        IF COMRAF(FAU(1), L, ST) = 1
            THEN FAULTD := I;
            LOOKFV(FAULTD, NOVAR, NAMF, NAMT);
            RETURN;
        END;
    END;
REP;
    TWRT("#NL NO SUCH FAULT HAS BEEN DEFINED (DESIGN,RTL)");
ENDPROC;
% DESCRIPTION OF THE STREAMS %

EXT PROC(REF BYTE, INT, RAB) GETBYT;
EXT PROC() REF STREAM NEXTFS;
EXT PROC(REF INT, INT, INT, INT, RAB, LABEL) GETINT;
EXT PROC(INT) REF UNITDE LOGAUN;
EXT PROC(REF BYTE, INT) SBITBY;
EXT PROC(REF UNITDE, REF STREAM) DESVAR;
EXT PROC(REF INT, INT) INT GETGIBI;
EXT PROC(REF STREAM) REF STREAM SETSTR;

EXT PROC DESTR(REF UNITDE U);
  REF STREAM S := NILSTM;
  BYTE IOS;
  INT INU, OUTU;
  GETBYT(IOS, 1, "I/O STREAM (1 = IN, 0 = OUT, 2 = DUMMY, ++ = TERMIN)");
  WHILE IOS # 1 DO
    IF IOS = '1'
      THEN L1: GETINT(INU, 1, MAXNOUNIT, 1, "FROM UNIT :");
      IF U.INST := NILSTM
        THEN S := NEXTFS();
        U.INST := S;
      ELSE S := U.INST;
      IF GETGIBI(S, S.I1O, 2) # 1
        THEN S := SETSTR(S);
      ELSEIF GETGIBI(S, S.I1O, 2) = 1
        AND S.FROMU.UNUM # INU
        THEN S := SETSTR(S);
      END;
      SBITBY(S, S.I1O, 1);
      S.TOUN := U;
      S.FROMU := LOGAUN(INU);
      DESVAR(U, S);
      ELSEIF IOS = '0'
        THEN L2: GETINT(OUTU, 1, MAXNOUNIT, 1, "TO UNIT :");
        IF U.OUTST := NILSTM
          THEN S := NEXTFS();
          U.OUTST := S;
        ELSE S := U.OUTST;
        IF GETGIBI(S, S.I1O, 2) # 1
          THEN S := SETSTR(S);
        ELSEIF GETGIBI(S, S.I1O, 2) = 1
          AND S.TOUN.UNUM # OUTU
          THEN S := SETSTR(S);
        END;
        SBITBY(S, S.I1O, 2);
        S.FROMU := U;
        S.TOUN := LOGAUN(OUTU);
        DESVAR(U, S);
        ELSEIF IOS = '2'
          THEN IF U.INST := NILSTM
            THEN U.INST := NILINT;
            ELSEIF U.OUTST := NILSTM
              THEN U.OUTST := NILOUT;
              ELSE
                TWRT("#NL# ERROR IN (" 
                "DESTR.RTL")");
            END;
          END;
        END;
      END;
    END;
  END;
END;
GETBYTE IOS, 1, "I/O STREAM(1-IN, @-OUT, 2: DUMMY, ++TERMIN)");
S::NEXTS := NILSTM;
REP;
ENDPROC;
% DESCRIPTION OF THE UNITS %

EXT PROC (REF UNITDE) DESTR;
EXT PROC (INT, INT, BYTE) PRICHA;

EXT PROC DESUN ();

INT K;

REF UNITDE U := NILUNI;
TWRT("#NL(3)#TOPOLOGY OF THE SYSTEM PART: 2");
PRICHA(1, 32, '*');
TWRT("#NL(2)#STREAMS AND VARIABLES OF EACH UNIT");
PRICHA(1, 32, '-');
TWRT("#NL(3)#");
FOR I := 1 TO COUNIT DO
  U := ONI(1);
  TWRT("#NL(3)#UNIT NO. :");
  IWR U(UNIT);
  K := U.UNUM;
  TWRT("#SP(#)");
  TWRT("TYU(K)*L");
  PRICHA(1, 23, '*');
  TWRT("#NL(2)#INPUT STREAMS");
  PRICHA(1, 13, '-');
  TWRT("#NL(2)#");
  DESTR(U);
  TWRT("#NL(2)#OUTPUT STREAMS");
  PRICHA(1, 14, '-');
  TWRT("#NL(2)#");
  DESTR(U);
  U.NEXU := NILUNI;
REP;
ENDPROC;
% DESCRIPTION OF VARIABLES %

TEXT PROC (REF BYTE, INT, RAM) GETBYT;
TEXT PROC (BYTE, INT) INT GETBIB;
TEXT PROC (REF UNITDE, REF STREAM) DATVAR;
TEXT PROC () REF VARIAB NEXTFU;
TEXT PROC () REF STREAM NEXTFS;
TEXT PROC (REF BYTE, INT) SBITBY;
TEXT PROC (REF STREAM, REF UNITDE) COMPVA;
TEXT PROC (REF STREAM) COMPVI;

ENTRY PROC DESVAR (REF UNITDE U, REF STREAM S);
BYTE ANS, VI;
L1: GETBYT(VI, 3, "VARIABLE DESCRIPTION(Y/N)");
WHILE VI = 'Y' DO
  IF GETBIB(S, SIG, 1) = 1
    THEN L2: GETBYT(ANS, 1, "IS THIS AN INTERNAL VAR.(Y/N)")
    IF ANS = 'Y'
      THEN DATVAR(U, S);
    ELSEIF ANS = 'N'
      THEN IF S.FROMU.OUTST := : NILSTM
      THEN BLOCK
        REF VARIAB U := NEXTFU();
        REF STREAM SA := NEXTFS();
       SA.FROMU := S.FROMU;
       SA.TOUN := S.TOUN;
       SA.NAVAR := U;
       SA.NEXTS := NILSTM;
       S.BITEY(SA.EIO, 2);
       S.FROMU.OUTST := SA;
       VI.NEXVAR := NILVAR;
       COMPVA(SA, U);
       COMPVI(SA);
      END BLOCK;
      ELSE COMPVA(S.FROMU.OUTST, U);
    END;
  END;
END REP;
ELSE IF GETBIB(S, SIG, 1) = 0
  THEN DATVAR(U, S);
ELSE TWRT("#NL#ERROR (DESVAR.RTL)"");
GOTO L2;
END;
ELSEIF GETBIB(S, SIG, 1) = 0
  THEN TWRT("#NL#ERROR IN INPUT (DESVAR" "RTL) PLEASE TYPE (Y/N)"");
GOTO L2;
END;
ENDFRQG;

PAGE DESVAR.RTL
EXT PROC(INT, INT, BYTE) PRICHAC;
EXT PROC(INT) PRINTN;

ENT PROC DISPLAY() INT:
    REF VARIAB V1 := NILVAR;
    REF DATAV D1 := NILDAT;
    BYTE FLAG := '0';
    INT K;
    PRICHAC(3, 26, SP);
    TWRT("MEASURED VARIABLES");
    PRICHAC(1, 26, SP);
    PRICHAC(0, 18, '*');
    PRICHAC(3, 8, SP);
    TWRT("VARIABLES#SP(3)#HI LIMIT#SP(3)#LO LIMIT#SP(3)#" 
        "STATUS#SP(3)#PRIORITY");
    PRICHAC(1, 8, SP);
    OUT('"');
    FOR I := 1 TO 3 DO
        PRICHAC(0, 8, '-');
        PRICHAC(0, 3, SP);
        REP;
        PRICHAC(0, 6, '-');
        PRICHAC(0, 3, SP);
        PRICHAC(0, 8, '-');
        TWRT("#NL"));
    FOR I := 1 TO MAXNOVAR DO
        V1 := VABC(1);
        IF V1.VALUES :=: NILDAT AND V1.MESU = 'Y'
            THEN D1 := V1.VALUES;
            OUT(V1.NAMEF);
            OUT(V1.NAMEE);
            PRINTN(V1.NOVA);
            PRICHAC(0, 8, SP);
            PRINTN(D1.HL);
            PRICHAC(0, 3, SP);
            PRINTN(D1.LL);
            PRICHAC(0, 7, SP);
            IF D1.ACTVA > D1.HL
                THEN TWRT("HI");
            ELSEIF D1.ACTVA < D1.LL
                THEN TWRT("LO");
            ELSE TWRT("OK");
            END;
            PRICHAC(0, 8, SP);
            K := D1.FRIO;
            PRINTN(K);
            FLAG := '1';
            END;
            REP;
            IF FLAG = '0'
                THEN PRICHAC(2, 18, SP);
                TWRT("** THERE ARE NO MEASURED VARIABLES IN THE" 
                    " SYSTEM");
                RETURN(0);
            END;
            RETURN(1);
            ENDPROC;

% DISPLAY MEASURED VARIABLES %
% ECHO INPUT FROM A DEVICE TO KB:

EXT PROC () BYTE RSXIN;

ENT PROC ECHOIN () BYTE;
    BYTE B := RSXIN();
    IF B ≠ LF
        THEN OUT(B);
        END;
    RETURN(B);
ENDPROC;

PAGE

ECHOIN.RTL
% FIND THE VARIABLE IN THE SYSTEM %

EXT PROC (BYTE, INT) INT GETBIB;

ENT PROC FIVIS(REF BYTE IO) REF VARIAB;
   REF STREAM SAU := NILSTM;
   REF STREAM SIO := NILSTM;
   REF VARIOB VIO := NILVAR;
   L1: IF GETBIB(IO, 3) = 1 OR GETBIB(IO, 1) = 1
       THEN SIO := UST INST;
           IF SIO.NEXTS #: NILSTM AND
               GETBIB(IO, 1) = 1
               THEN SAU := SIO.NEXTS;
           END;
       ELSEIF GETBIB(IO, 1) = 0
           THEN SIO := UST OUTST;
       END;
       WHILE SIO #: NILSTM DO
           IF SIO.NAVAR #: NILVAR OR GETBIB(IO, 1) = 1
               THEN SIO := SIO.FROMU OUTST;
               WHILE SIO #: NILSTM DO
                   IF SIO.TOUN #: UST
                       THEN SIO := SIO.NEXTS
                                ELSE GOTO L2;
                           END;
           END;
           REP;
       END;
   END;
   L2: WHILE SIO #: NILSTM DO
       VIO := SIO.NAVAR;
       WHILE VIO #: NILVAR DO
           IF VIO.NAMEF = EST VAR OR
               VIO.NAMET = EST VAR
               THEN RETURN(VIO);
           END;
           REP;
       VIO := VIO.NEXVAR;
       REP;
       SIO := SIO.NEXTS;
       REP;
       SIO := SAU;
       IF SAU #: NILSTM
           THEN SAU := SAU.NEXTS;
       END;
       REP;
       TWR('<"NIL UNDEFINED VARIABLE IN THE SYSTEM (FIVIS.RTL)">);
       OUT(EST VAR);
       EST := EST.NEXT;
       IF EST.IDEN = 'M' OR EST.IDEN = SP
           THEN RETURN(NILVAR);
       END;
       IF EST.IDEN # 'R' AND EST.IDEN # 'B'
           THEN IO := EST.INOU;
               SIO := NILSTM;
               SAU := NILSTM;
               VIO := NILVAR;
               GOTO L1;
           END;
       RETURN(NILVAR);
   ENDPROC;
OPTION(1)BC,TR;
TITLE
---------------------
RTL "RESIDENT LIBRARY
---------------------

LET NC = OCT 15, OCT 12;

EXT PROC (INT) IREAD;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;

EXT PROC (INT) IWRT;
EXT PROC (REF ARRAY BYTE) TWRT;

EXT PROC (INT, INT) IWRTF;

EXT PROC (INT) NLS, SFS;

ENT PROC NCLS (INT N);
  FOR J := 1 TO N DO
    TWRT("#NC#");
    REP;
  RETURN;
ENDPROC;
EXT PROC (REF EVENT, REF GATE) REF EVENT DELEVT;
EXT PROC (REF EVENT, REF GATE) REF GATE DELGAT;

ENT PROC GARCOL (){
    REF GATE G := GST;
    REF EVENT E := NILEV; E1 := NILEV;
    GST := GST.BACKE.BACKG;
    BCST := GST.BC;
    IF GST ::= NILGA THEN RETURN; END;
    WHILE G ::= GST DO
        IF G.NEXTG ::= NILEV
            THEN IF G.NEXTG.UNNO ::= NILUNI
            THEN E := G.NEXTG;
            END;
        WHILE E.NEXTG ::= NILGA AND
            E.UNNO ::= NILUNI DO
            E := DELEVT(E, G);
        END;
        IF E ::= NILUNI
            THEN IF E.UNNO ::= NILUNI THEN GOTO L1; END;
                G := E.NEXTG;
                E := DELEVT(E, G);
                EST := E;
        END;
    END;
    L1: E := G.BACKE;
    G := DELGAT(E, G);
    E := DELEVT(E, G);
    EST := E;
    L2: REP:
        E1 := GST.NEXTE;
        IF E1.UNNO ::= NILUNI
            THEN WHILE E1.NEXT.UNNO ::= NILUNI DO
                E1 := E1.NEXT;
            END;
            REP;
        ELSE EAST := E1;
            EAST.NEXT := EST;
        ELSE
            EAST := GST.BACKE;
            GST.NEXTE := NILEV;
            G := '+';
        END;
    END;
ENDPROC;
% GET A BIT IN A BYTE %

ENT PROC GETBI(BYTE BYT, INT BITNUM) INT;
    RETURN((BYT SLL (8-BITNUM)) SRL 7);
ENDPROC;

-------
ENT PROC GETBIT(INTEGER WORD, BITNUM) INTEGER;
RETURN((WORD SLL (16-BITNUM)) SRL 15);
ENDPROC;
% GET BYTE %

EXT PROC(RAB, INT) OUTP;

INT PROC GETBYTE(REF BYTE B, INT LFEED, RAB AB);
OUTP(AB, LFEED);
BLOCK
  BYTE BB := IN();
  WHILE BB = CR OR BB = LF DO
    BB := IN();
  END;
  VAL B := BB;
ENDBLOCK;
ENDPROC;
% GET FRACTION %

EXT PROC (REF ARRAY BYTE, INT) OUTP;
EXT PROC () FRAC FREAD;
EXT PROC (FRAC, FRAC, LABEL) TESTF;

ENT PROC GETFRAC(REF FRAC F, FRAC LO, INT LFEED, RAB AB, LABEL ERRLAB);
   FRAC A := 0.0B0;
   OUTP(AB, LFEED);
   A := FREAD();
   TESTF(A, LO, ERRLAB);
   VAL F := A;
ENDPROC;

PAGE GETFRAC.RTL
% GET AN INTEGER %

EXT PROC (REF ARRAY BYTE, INT) OUTP;
EXT PROC (INT, INT, INT, LABEL) TESTI;

INT PROC GETINT (REF INT I, INT LO, HI, LFEED, REF ARRAY BYTE AB,
                   LABEL ERRLAB);

    INT T := 0;
    OUTP(AB, LFEED);
    T := IREAD();
    TESTI(T, LO, HI, ERRLAB);
    VAL I := T;
    RETURN;
ENDPROC;

PAGE
----
GETINT.RTL
% GET STRING %

EXT PROC(RAB, INT) OUTP;

ENT PROC GETSTR (REF ARRAY BYTE TARGET, BYTE TERMIN,
INT LFEED, RAB AB, LABEL ERRLAB);

OUTP(AB, LFEED);
FOR J := 1 TO LENGTH TARGET DO
BLOCK
  BYTE B := SP;
L1: B := IN();
  IF B = CR OR B = LF THEN GOTO L1; END;
  IF B = TERMIN
  THEN FOR K := J TO LENGTH TARGET DO
    TARGET(K) := SP;
  REP;
  RETURN;
  ELSE TARGET(J) := B;
  END;
ENDBLOCK;
REP;
TWRT("$NL,LF$BYTE ARRAY FULL$NL,LF$);
GOTO ERRLAB;
ENDPROC;
% LOCATE THE 1ST ELEMENT OF THE MINITREE %

ENT PROC LFEMT ();
  FOR I := 1 TO MAXNUNIT DO
    IF LOT(I).TYPE = UST.TY
      THEN EST := LOT(I).FIRSEL;
      RETURN;
    END;
  REPT:
  TWR(("UNIT TYPE OF UNIT UNDEFINED (LFEMT.RTL)");
ENDPROC;
% LINK THE ARRAYS USED IN THE TOPOLOGY OF THE SYSTEM %

EXT PROC(REF ARRAY STREAM) LS;
EXT PROC(REF ARRAY UNIDEX) LU;
EXT PROC(REF ARRAY VARIA) LV;

ENT PROC LINTOP ();
    LS(STR);
    LU(UNI);
    LV(VAB);
ENDPROC;
OPTION(1) EC, TR;
TITLE
LINK TOPOLOGY;

LET COMERR = 5000;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC() LINTOP;
EXT PROC() RSXINT;
EXT PROC(IN) RSXCL;
EXT PROC(IN, REF ARRAY BYTE) RSXOP1, RSXOPO;
EXT PROC(IN) INT RSXSWI, RSXSWO;
EXT PROC(REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC() BYTE COMIP;

SVC DATA RRERR:
   LABEL ERL;
   INT ERN;
   PROC(IN) ERP;
ENDDATA;

SVC DATA RRSIO:
   PROC() BYTE IN;
   PROC(BYTE) OUT;
ENDDATA;

SVC DATA RRERRX;
   INT LINENO;
   BYTE UERL, ERRUN;
   INT RSXDSW;
ENDDATA;

DATA FLDATA:
   ARRAY (35) BYTE IFILE := "$$SP(35)$$";
   ARRAY (35) BYTE OFILE := "$$SP(35)$$";
ENDDATA;

ENT PROC RRJOB();
   ERRUN := 3;
   IN := COMIP;
   IF IN(0) # 'L' OR IN(0) # 'I' OR IN(0) # 'T' OR IN(0) # ' ' THEN ERP(COMERR);
   END;

   TREAD(OFILE, "=");
   TREAD(IFILE, "$$LF$$");
   RSXINT();
   RSXOP0(1, OFILE);
   RSXSWO(1);
   RSXOP1(2, IFILE);
   RSXSWI(2);
   LINTOP();
   RSXCL(1);
   RSXCL(2);
ENDPROC;
% LOCATE UNITS %

ENT FROC LOGCAUN (INT IO) REF UNITDEJ
  FOR I := 1 TO COUNT DO
    IF UNI(I) • UNUM = IO
    THEN RETURN(UNI(I));
  END;
  REP;
  TWRT("NO SUCH UNIT IN THE SYSTEM (LOGCAUN.RTL)");
  RETURN(NILUNI);
ENDPROC;
% LOOKING FOR THE VARIABLE %

EXT PROC (INT) LORIMI;

ENT PROC LOOKFV (INT FAULTD, NOVAR, BYTE NAMF, NAMT);
    REF VARIAB V := NILVAR;
    FOR I := 1 TO MAXNOVAR DO
        V := VAB(I);
        IF V$NAMEF = NAMF AND
            V$NAMET = NAMT
            THEN IF V$NOVA = NOVAR
                THEN VST := V;
                LORIMI(FAULTD);
                RETURN;
        END;
    END;
    REP;
    TWRT("NIL NO SUCH VARIABLE IN THE SYSTEM(LOOKFV$RTL)" );
ENDPROC;
% LOCATE THE RIGHT MINITRE %

EXT PROC (BYTE, INT) INT GETBIB;
EXT PROC () LFMT.BULD();
EXT PROC (BYTE, BYTE, INT) INT LRUMT;

ENT PROC LORIMII (INT FAULTD);
    IF BT = 0 THEN GOTO L; END;
    IF EST.IDEN = 'R'
        THEN LFEMT();
            IF LRUMT(2, SP, FAULTD) = 1
                THEN GOTO L2;
            ELSE TWR("No SUCH R EVENT IN UNIT NO.");
                TWR(UST.UNUM);
                TWR(" LORIMI:RTL");
                TWR("GR");
                RETURN;
        END;
END;
L: 
    IF VST := NILVAR THEN GOTO L1; END;
    SST := VST.BACKS;
    UST := SST.FROMU;
    IF VST.NAMEF $ SP AND VST.NAME = SP AND SST.SIO = 1
        THEN UST := SST.TOUN;
        LFEMT();
        IF LRUMT(3, VST.NAMEF, FAULTD) = 1
            THEN GOTO L2;
    END;
L1: TWR("No ERROR INTERNAL VAR. LORIMI:RTL#CR");
    RETURN;
END;
LFEMT();
    IF LRUMT(0, VST.NAMEF, FAULTD) # 1
       THEN UST := SST.TOUN;
       LFEMT();
       IF LRUMT(1, VST.NAMEF, FAULTD) # 1
           THEN TWR("No ERROR (LORIMI:RTL)");
           RETURN;
END;
L2: BULD();
ENDPROC;
ENT PROC LRAB(RAB A, B);
    IF LENGTH B > LENGTH A THEN GOTO L1; END;
    FOR I := 1 TO LENGTH B DO
        A(I) := B(I);
    END;
    FOR J := LENGTH B + 1 TO LENGTH A DO
        A(J) := SP;
    END;
    RETURN;
L1: WRITE("STRING TOO LONG");
ENDPROC;
% LOCATE THE RIGHT VARIABLE IN THE MINITREE %

EXI PROC (BYTE, INT) INT GETBIB;

EXT PROC LRVMT (BYTE IO, NAM, INT FAULTD) INT;

WHILE EST #: NILEO DO
    IF EST VAR = NAM AND EST FAULT = FAULTD
        THEN IF IO #: 0
            THEN IF GETBIB (EST INQU, IO) = 1
                THEN RETURN (1);
            END;
        ELSEIF GETBIB (EST INQU, 1) = 0
            THEN RETURN (1);
        END;
    END;
    EST := EST TOP;
    REPI;
    RETURN (0);
ENDPROC;
% LINK STREAMS %

ENT PROC LS(REF ARRAY STREAM S);
FOR I := 1 TO MAXNOSTRM1 DO
  S(I).NEXTS := S(I+1);
REP;
ENDPROC;
% LINK UNITS %

ENT PROC LU(REF ARRAY UNITDE U);
  FOR I:= 1 TO MAXNOUNI THI DO
    U(I) : NEXU := U(I+1); 
  REP;
ENDPROC;
ENT PROC LV(REF ARRAY VARIAB V);
    FOR I := 1 TO MAXNOVARM DO
        V(I) = EXVAR := V(I+1);
    END浦;
ENDPROC;
DATA OF A 'M' EVENT

EXT PROC(REF EVENT) SGA(E)

ENT PROC MEVT(REF EVENT E)
  SGA(E))
ENDPROC
EXT PROC () REF EVENT NEXTFE;
EXT PROC (INT) LORIMI;
EXT PROC (REF BYTE, INT) SHITBY;

ENT PROC NEWEVU();
  REF EVENT E := NEXTFE();
  E.IDEN := EST.IDEN;
  E.UNID := EST.UNID;
  E.VAR := EST.VAR;
  E.INOU := EST.INOU;
  IF E.IDEN $ 'B' OR E.IDEN $ 'R'
  THEN E.NAVA := VST;
  SHITBY(VST, EX, 2);
END;

E.FAULT := EST.FAULT;
E.BACKG := GST;
E.UNNO := UST;
E.PROB := EST.PROB;
IF EST.NEXT.IDEN $ 'M' OR EST.NEXT.IDEN $ SP
  THEN E.NEXT := NILEV;
ELSE E.NEXT := EST.NEXT;
END;

IF C $ '*'
  THEN EAST.NEXT := E;
      C := '+';
END;

EAST := E;
IF GST.NEXTG := NILEV
  THEN GST.NEXTE := E;
END;

IF EST.NEXTG := QM
  THEN LORIMI(E.FAULT);
END;
ENDPROC;
ENT PROC NEXTFB () REF BOUNDEC0;
IF FREEBC := NILNEBC THEN TWR("NL#NO MORE B. C. AVAILABLE");
RETURN(NILNEBC);
END;

  BLOCK
    REF BOUNDEC0 I := FREEBC;
    FREEBC := I\#NEXTBC;
    RETURN(I);
  ENDBLOCK;
ENDPROC;
% GET THE NEXT EVENT %

ENT PROC NEXTFE() REF EVENT:
  IF FREEV := NILNEX THEN TWRT("#NL# NO MORE EVENTS AVAILABLE");
  RETURN(NILNEX);
END:
  BLOCK
    REF EVENT I := FREEV;
    FREEV := I*NEXT;
    RETURN(I);
  ENDBLOCK;
ENDPROC;
% GET THE NEXT GATE %

ENT PROC NEXTFG() REF GATE;
    IF FREEGA := NILNEG THEN Twrt("#NL#NO MORE GATES AVAILABLE");
    RETURN(NILNEG);
END;
    BLOCK
    REF GATE I := FREEGA;
    FREEGA := I NNEGATE;
    RETURN(I);
    ENDBLOCK;
ENDPROC;
ENT PROC NEXTFS(); REF STREAM;
   IF FREEST ::: NILSTN
   THEN TWRIT("#NL# NO MORE STREAMS AVAILABLE IN ARRAY ");
   END;
END BLOCK
   REF STREAM S := FREEST;
   FREEST := S.NEXTS;
   RETURN(S);
END BLOCK;
ENDPROC;

% GET NEXT FREE STREAM %
ENT PROC NEXTFU() REF UNITDE;
  IF FREEUN :=: NILUNE
  THEN TWRT("NL$ NO MORE UNITS AVAILABLE IN ARRAY");
  END;
  BLOCK
  REF UNITDE U := FREEUN;
  FREEUN := U.NEXUS;
  RETURN(U);
  ENDBLOCK;
ENDPROC;

% GET NEXT FREE UNIT %
% GET NEXT FREE VARIABLE %

ENT PROC NEXTFV() REF VARIAB;
IF FREEVA := NILVAN
THEN TWRN("NO MORE VARIABLES AVAILABLE IN ARRAY ");
END;
BLOCK
  REF VARIAB V := FREEVA;
  FREEVA := V.NEXVAR;
  RETURN(V);
ENDBLOCK;
ENDPROC;
% NO NEED TO DEVELOP AN EVENT AGAIN %

EXT PROC (...) REF EVENT NEXTF();
EXT PROC (REF BYTE, INT) SBITBY();

ENT PROC NODEVL (REF EVENT E);
REF EVENT E1 := NEXTF();
E1.IDEN := EST.IDEN;
E1 UNID := EST.UNID;
E1.VAR := EST.VAR;
E1.INOU := EST.INOU;
SBITBY(E1.INOU,B);
E1.NAVA := VST;
E1.FAULT := EST.FAULT;
E1.BACKG := GST;
E1.NEXTG := E.NEXTG;
E1.UNNO := UST;
E1.PROB := EST.PROB;
IF C = 'x'
    THEN EAST.NEXT := E1;
        C := '+';
END;
EAST := E1;
IF GST.NEXT := NILEV
    THEN GST.NEXT := E;
END;
IF EST.NEXT.IDEN = 'm' OR EST.NEXT.IDEN = SP
    THEN E1.NEXT := NILEV;
    ELSE E1.NEXT := EST.NEXT;
END;
ENDPROC;
% PRINT A RAB AND PROMPT %

EXT PROC (INT) NCLS;

ENT PROC OUTP (REF ARRAY BYTE AB, INT LFEED);
  NCLS(LFEED);
  TWRT(AB);
  TWRT("#EQ#");
ENDPROC;

OPTION(1) BC, TR;
TITLE

PRINT MINITREES;
LET COMERR = 5022;
LET SP = OCT 40;
LET LF = OCT 121

EXT PROC READNU;
EXT PROC INT RSXINT;
EXT PROC INT RSXCL;
EXT PROC INT RSXPOI, RSXPO;
EXT PROC INT INT RSXSWI, RSXSWO;
EXT PROC INT REF ARRAY BYTE INT TREAD;
EXT PROC INT BYTE COMIP;

SVC DATA RRERR;
LABEL ERL;
INT ERN;
PROC INT ERP;
ENDDATA;

SVC DATA RRSIO;
PROC INT BYTE IN;
PROC BYTE OUT;
ENDDATA;

SVC DATA RRERRX;
INT LINENO;
BYTE UEF I GE, ERLUN;
INT RSXDSW;
ENDDATA;

DATA FILDAT;
ARRAY (35) BYTE IFILE := "#SP(35)#";
ARRAY (35) BYTE OFILE := "#SP(35)#";
ENDDATA;

EXT PROC RRJOB();
ERRLUN := 3;
IN := COMIP;
IF IN() # 'E' OR IN() # 'M' OR IN() # 'T' OR IN() # '
THEN ERP(CC(MERR));
END;
TREAD OFILE,"=";
TREAD IFILE,"##LF##";
RSXINT();
RSXPOI, OFILE);
RSXSWO(1);
RSXPOIC2, IFILE);
RSXSWI(2);
READNU();
RSXCL(1);
RSXCL(2);
ENDPROC;
% PRINT HEADS FOR MINITREES %

EXT PROC (INT, INT, BYTE) PRICHA;

ENT PROC PRHEMT ();
PRICHA(2, 16, SP);
TWRT("VARIABLE");
PRICHA(0, 29, SP);
TWRT("B.C & NOT");
PRICHA(1, 15, SP);
TWRT("DESCRIPTION$SP(6)$FAULT$SP(8)$GATE$SP(4)$ALLOWED FAULTS");
PRICHA(1, 15, SP);
PRICHA(0, 11, '-');
PRICHA(0, 6, SP);
PRICHA(0, 5, '-');
PRICHA(0, 6, SP);
PRICHA(0, 4, '-');
PRICHA(0, 6, SP);
PRICHA(0, 13, '-');
TWRT("#NL#");
ENDPROC;

PAGE

PRHEMT. RTL
OPTION(1) BC, TR;
TITLE
PRINT TOPOLOGY OF THE SYSTEM;

LET COMERR = 5000;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC(FRIUN);
EXT PROC(RSXINT);
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOPI, RSXOPO;
EXT PROC(INT) INT RSXSWI, RSXSWO;
EXT PROC(REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC() BYTE COMIF;

SVC DATA RRSIO;
PROC() BYTE IN;
PROC(BYTE) OUT;
ENDDATA;

SVC DATA RRSIO;
PROC() BYTE IN;
PROC(BYTE) OUT;
ENDDATA;

SVC DATA RRSIO;
INT LINENO;
BYTE UEFLAG, ERLUN;
INT RSXDSW;
ENDDATA;

DATA FILDAT;
ARRAY(35) BYTE IFILE := "#SF(35)#";
ARRAY(35) BYTE OFILE := "#SF(35)#";
ENDDATA;

ENT PROC RRJOB();
ERLUN := 3;
IN := COMIF;
IF IN() # "P" OR IN() # "R" OR IN() # "I" OR IN() # " "
   THEN ERP(COMERR);
END;

TREAD(OFILE,"=");
TREAD(IFILE,"#LF#");
RSXINT();
RSXOFO(1, OFILE);
RSXSWO(1);
RSXOPI(2, IFILE);
RSXSWI(2);
FRIUN();
RSXCL(1);
RSXCL(2);
ENDPROC;

PAGE
% PRINT ALL TREES %

EXT PROC () PRTREE;

INT PROC PRAILL();
INT I := CI;
REF EVENT E := EVT(CI);
WHILE E.UNNO #: NILUNI DO
  IF E.IDEN = "M"
    THEN EAST := E;
      PRTREE();
    END;
  I := I + 1;
  E := EVT(I);
END;
ENDPROC;

% PRINT B, C, AND N, A, C FOR MINITREE %

EXT PROC (INT, INT) INT GETBIT;
EXT PROC (INT, INT, BYTE) PRICHA;
EXT PROC (INT) TRNOWR;

ENT PROC PRI BNA (INT B, N, S, N) INT;
    INT I := 1, CB := 0;
    WHILE NS = 0 AND I <= 16 DO
        IF GETBIT(B, I) = 1
            THEN NS := 1;
            PRICHA(0, 3, SP);
            CB := 16*(N-1);
            TRNOWR(I+CB);
        END;
        I := I+1;
    REP;
    FOR J := I TO 16 DO
        IF GETBIT(B, J) = 1
            THEN PRICHA(I, 53, SP);
            CB := 16*(N-1);
            TRNOWR(J+CB);
        END;
    REP;
    RETURN(NS);
ENDPROC;
% PRINT CHARACTERS %

EXT PROC(INT) NCLS;

INT PROC PRICHA (INT LFEED, INT N, BYTE CHAR);
NCLS(LFEED);
FOR I := 1 TO N DO
  OUT(CHAR);
REP;
ENDPROC;
% PRINT FAULT ACCORDING TO THE BIT SET %

EXT PROC(INT,INT) INT GETBIT;
EXT PROC (INT) TRNWR;
EXT PROC (INT,INT,BYTE) PRICHA;

INT PROC PRIFFB (INT B ,INT BINU);
    INT CB := 0;
    FOR I := 1 TO 16 DO
        IF GETBIT(B,I) = 1
            THEN PRICHA(I,5,SP);
                CB := 16 * (BINU-1);
                TRNWR(I+CB);
            END;
    REPE;
ENDPROC;
% PRINT HEADS %

EXT PROC (INT, INT, BYTE) PRICHA;

INT PROC PRIHE6 ( );
PRICHA(2, 12, SP);
TWRT("NAME NO.");
PRICHA(1, 12, SP);
TWRT("VAR. VAR. FAULT" IFSP(6) DESCRIPTION OF UNIT SP(5) ;"
"GATE") ;
PRICHA(1, 12, SP); PRICHA(0, 4, '=');
OUT(SP); PRICHA(0, 4, '=');
OUT(SP); PRICHA(0, 5, '=');
PRICHA(0, 6, SP); PRICHA(0, 19, '=');
PRICHA(0, 5, SPY; PRICHA(0, 4, '=');
TWRT("#NL#");
ENDPROC;
% PRINT MMiniTREES %

EXT PROC (INT, INT, BYTE) PRICHA;
EXT PROC (RAB) TWRAE;

ENT PROC PRIMNT (INT N);
  INT N1;
  N1 := 'N';
  PRICHA(3, 68, ' * ');
  PRICHA(2, 26, SP);
  PRICHA(0, 13, ' * ');
  PRICHA(1, 26, SP);
  TWRT("** MMiniTREES **");
  PRICHA(1, 26, SP);
  TWRT("** FOR **");
  PRICHA(1, 26, SP);
  OUT('*');
  N := 1 + (9-N)/2;
  PRICHA(8, N, SP);
  TWRAE(ST);
  N := 11-N-N1;
  PRICHA(0, N, SP);
  OUT('*');
  PRICHA(1, 26, SP);
  PRICHA(0, 13, ' * ');
ENDPROC;

EXT PROC (BYTE, INT) INT GETBIB;
EXT PROC (INT, INT, BYTE) PRICHA;
EXT PROC (INT, INT, INT) INT PRIENA;
EXT PROC (INT) TRNWR;

EXT PROC PRINAC ();
    REF BOUNDEC B := GST·BC;
    INT NS := Ø;
    IF GETBIB(GST·CLASS, 2) = 1
        THEN GOTO L1;
        ELSEIF GETBIB(GST·CLASS, 1) = 1
            THEN PRICHA(B, 2, SP);
            ELSE PRICHA(B, 3, SP);
        END;
    L1: NS := PRIENA(B·NA1, NS, 1);
        NS := PRIENA(B·NA2, NS, 2);
        NS := PRIENA(B·NA3, NS, 3);
    ENDPROC;
% PRINT NOMENCLATURE %

EXT PROC (INT, INT, BYTE) PRICHAC;

ENDPROC;
% PRINT INTEGER WITH SPACES %

ENT PROC PRINTN(INTEGER N);
    IF N < 10
        THEN TWRT("#SP(2)#");
    ELSEIF N < 100
        THEN OUT(SP);
    END;
    IWRT(N);
ENDPROC;

% PRINT VARIABLES %

ENT PROC PRINVA(REF VARIAB V);
  WHILE V :#: NILVAR DO
    OUT(V.NAME);
    OUT(V.NAME);
    IWRT(V.NOVA);
    IF V.MESU = 'Y' THEN IWRT(" (M) ");END;
    IF V.NEXVAR :#: NILVAR THEN OUT('.');END;
    V := V.NEXVAR;
  REP;
ENDPROC;
% PRINT STREAMS %

EXT PROC(REF STREAM) CHEVBPS;

ENT PROC PRIST (REF UNITDE U, BYTE IO);
   REF STREAM SU := NILSTM;
   IF IO = '1'
      THEN SU := U.INS;
      ELSE SU := U.OUTS;
   END;
   WHILE SU !# NILSTM DO
      IF SU !# NILINS OR SU !# NILOUT
      THEN TWRT("\#NL\#NONE");
         RETURN;
      END;
      TWRT("\#NL\#FROM : ");
      IWRT(SU.FROM.U.UNUM);
      TWRT("\#NL\#TO : ");
      IWRT(SU.TO.U.UNUM);
      CHEVBPS(SU);
      SU := SU.NEXTS;
   END;
   REP;
ENDPROC;
% PRINT THE TREE %

EXT PROC (INT, INT, BYTE) PRICHAS;
EXT PROC () PRTRS, PRINOM;

EXT PROC PRITRE ();
  PRINOM();
  TWRT("\%NL(2)\%"");
  PRICHAS(0, 24, SP);
  TWRT("### FAULT TREE ###");
  PRICHAS(1, 24, SP);
  PRICHAS(0, 18, '=');
  PRTRS(); 
ENDPROC;

% PRINT TYPE OF UNIT %

ENT PROC PRITYU (BYTE B);
    INT I := B;
    Twrt(TYU(1),L);
ENDPROC;

PAGE

PRITYU.RTL
% PRINT UNITS %

EXT PROC(BYTE) PRIYUJ;
EXT PROC(REF UNITDE, BYTE) PRIYST;
EXT PROC(INT, INT, BYTE) PRICHAJ;

EXT PROC PRIUN ();
REF UNITDE UA := NILUNIT;
TWRT("#NL#TOPOLOGY OF THE SYSTEM");
PRICHAJ(1, 22, '*');
FOR I := 1 TO COUNT DO
  UA := UNIT(I);
  PRICHAJ(3, 65, '*');
  TWRT("#NL#UNIT NO. :");
  TWRT(UA.UNIT);
  PRICHAJ(1, 12, '-');
  TWRT("#NL#TYPE OF UNIT :");
  PRIYUJ(UA.TY);
  TWRT("#NL### INPUT STREAMS **");
  PRIST(UA. '1');
  TWRT("#NL### OUTPUT STREAMS **");
  PRIST(UA. '0');
  PRICHAJ(1, 65, '*');
REPI;
ENDPROC;
% PRINT VARIABLES OF MINITREES %

EXT PROC (BYTE, INT) INT GETBIE;
EXT PROC (INT, INT, BYTE) PRICHA;
EXT PROC (INT) TRNOWR;
EXT PROC () PRTGA;

EXT PROC PRIVAM (REF EVENT E);
  REF GATE G := NILGA;
  PRICHA(0, 4, SP);
  OUT(E, VAR);
  PRICHA(0, 4, SP);
  IF GETBIE(E, INOU, 1) = 1 THEN TWR(T:"IN");
      OUT(SF);
  ELSEIF GETBIE(E, INOU, 1) = 0 THEN TWR(T:"OUT");
  END;
  IF GETBIE(E, INOU, 2) = 1 OR GETBIE(E, INOU, 3) = 1 THEN PRICHA(0, 3, SP);
  END;
  PRICHA(0, 6, SP);
  TRNOWR(E, FAULT);
  IF E, IDEN = 'M' THEN PRICHA(0, 2, SP):
      G := GST;
      GST := E, NEXTG;
      PRTGA();
      PRIVAM();
      GST := G;
  END;
ENDPROC;
% PRINT VARIABLE OF TREE %

EXT PROC (INT) TRNOWR;
EXT PROC (INT, INT, BYTE) PRICHA;
EXT PROC (BYTE) PRIYU;

ENT PROC PRIVAT (REF EVENT E);
  IF E.NAVA #1 NILVAR
    THEN OUT(SP);
        OUT(E.NAVA.NAMEF);
        OUT(E.NAVA.NAMET);
        OUT(SP);
    IF E.NAVA.NOV < 10 THEN OUT(SP);END;
    IF E.NAVA.NOV < 100 THEN OUT(SP);END;
    IWRTE(E.NAVA.NOVA);
    PRICHA(0, 2, 5F);
    ELSE PRICHA(0, 10, SF);
    END;
    TRNOWR(E.FAULT);
    TWRT("UNIT NO. ");
    IF E.UNNO.UNUM < 10 THEN OUT(SP);END;
    IWRTE(E.UNNO.UNUM);
    PRICHA(0, 3, SP);
    PRIYU(E.UNNO.TY);
END PROC;

% PRINT GATE OF TREE %

EXT PROC (BYTE, INT) INT GETBIBJ

ENT PROC PRTGA();
  OUT(SP);
  IF GETBI (GST.CLASS, 2) = 1
      THEN TWRT("EX-OR");
  ELSEIF GETBI (GST.CLASS, 1) = 1
      THEN TWRT("AND");
  ELSE TWRT("OR");
  END;
ENDPROC;

PAGE

PRTGA .RTL
EXT PROC (REF EVENT) PRIVAT;
EXT PROC (BYTE, INT) INT GETBIB;
EXT PROC (INT, INT, BYTE) PRICH;
EXT PROC () PRTG;
PRHED;

EXT PROC PRTRS ();
REF EVENT E := NILEV;
WHILE EAST #: NILEV DO
  IF EAST.IDEN = 'B' THEN GOTO L1: END;
  TWRT("#NL(2)#");
  PRIHED();
  IF EAST.IDEN = 'M'
  THEN TWRT("TOP EVENT : ");
  ELSE PRICH(0, 4, SP);
  TWRT("EVENT : ");
END;
PRIVAT(EAST);
IF EAST.NEXTG #: NILGA AND GETBIB(EAST.INOU, B) = 0
THEN GST := EAST.NEXTG;
PRTG();
TWRT("#NL#");
E := GST.NEXTE;
WHILE E #: NILEV DO
  PRICH(1, 2, SP);
  OUT(E.IDEN);
  TWRT("-EVENT : ");
  PRIVAT(E);
  E := E.NEXT;
REP;
EAST := GST.NEXTE;
GOTO L2:
ELSEIF EAST.IDEN # 'B'
THEN IF GETBIB(EAST.INOU, B) = 1
  THEN PRICH(1, 32, SP);
  TWRT("EVENT ALREADY "
      "DEVELOPED*");
  ELSE PRICH(1, 33, SP);
  TWRT("*DIAMOND EVENT*"));
END;
IF EAST.NEXT #: NILEV
THEN EAST := EAST.NEXT;
GOTO L2;
END;

L1: WHILE EAST.IDEN = 'B' AND
EAST.NEXT #: NILEV DO
  EAST := EAST.NEXT;
REP;
IF EAST.NEXT #: NILEV AND EAST.NEXTG #: NILGA
THEN WHILE GST.BACKE.NEXTG #: NILGA
  AND GST.BACKE.BACKG #: NILGA
  DO GST := GST.BACKE.BACKG;
REP;
IF GST.BACKE.IDEN = 'M'
THEN EAST := GST.BACKE;
RETURN;
END;
EAST := GST.BACKE.NEXT;
GST := EAST.BACKG;
END;
L2: REP;
ENDPROC;
OPTION(1) BC, TR
TITLE
PRINT THE TREE

LET COMERR = 5000;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC( ) PRIALL;
EXT PROC( ) RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOPI, RSXOP0;
EXT PROC(INT) INT RSXSWI, RSXSWO;
EXT PROC(REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC( ) BYTE COMIP;

SVC DATA RRERR;
LABEL ERL;
INT ERR;
PROC(INT) ERP;
ENDDATA;

SVC DATA RRSIO;
PROC() BYTE IN;
PROC(BYTE) OUT;
ENDDATA;

SVC DATA RRERRX;
INT LINENO;
BYTE UEFLAG, ERRLUN;
INT RSXDSW;
ENDDATA;

DATA FILDAT;
ARRAY(35) BYTE IFILE := "#SP(35)#";
ARRAY(35) BYTE OFILE := "#SP(35)#";
ENDDATA;

ENT PROC RRJOB();
ERRLUN := 3;
IN := COMIP;
IF IN() # 'P' OR IN() # 'T' OR IN() # 'R' OR IN() # 'I' THEN ERP( COMERR);
END;

TREAD(OFILE, "=");
TREAD(IFILE, "#LF#"");
RSXINT();
RSXOPO(1, OFILE);
RSXSWO(1);
RSXOP1(2, IFILE);
RSXSWI(2);
PRIALL();
RSXCL(1);
RSXCL(2);
ENDPROC;
% TEST RANGE OF A FRACTION %

EXT PROC (FRAC, INT) FWRTF;
EXT PROC (INT) RNGEPR;

ENT PROC RANGEF (FRAC 1, INT HIORLO, LABEL ERRLAB);
   RNGEPR(HIORLO);
   FWRTF(1, 6);
   GOTO ERRLAB;
ENDPROC;
% TEST RANGE OF INTEGER %

EXT PROC (INT, INT) IWRTF;
EXT PROC (INT) RNGEPR;

ENT PROC RANGEP (INT HI, LO, LABEL ERRLAB);
    RNGEPR(HI);
    IWRTF(1, 6);
    GOTO ERRLAB;
ENDPROC;
% PROC USED IN GETRNO %

EXT PROC (INT) RNGEPR;
EXT PROC (REAL, INT, INT) RWRTF;

ENT PROC RANGER (REAL R, INT HIORLO, LABEL ERRLAB);
  RNGEPR(HIORLO);
  RWRTF(R, 4, 6);
  GOTO ERRLAB;
ENDPROC;
% READ NAME OF UNIT TO PRINT ITS MINITREES %

EXT PROC (INT, INT, BYTE) PRICHA;
EXT PROC () INT RETYUN;
EXT PROC (INT) PRIMNT;
EXT PROC () PRHEMT;
EXT PROC (REF EVENT) PRIVAM;

EXT PROC READNU ();
  REF EVENT E := NILEV;
  INT I := 1;
  INT B;
  TWRT("#NL(4)# READING TYPE OF UNIT TO PRINT MINITREES");
  PRICHA(1, 39, '=');
  TWRT("#NL(2)#");
  B := RETYUN();
  IF B = 0 THEN RETURN; END;
  WHILE ST(I) # SP DO
    I := I + 1;
  REP;
  PRIMNT(I-1);
  FOR J := 1 TO MAXNOUNIT DO
    IF LOT(J) . TYPE = B
      THEN E := LOT(J) . FIRST;
      GOTO L1;
    END;
  END;
L1: WHILE E . UNID = B DO
    IF E . IDEN = 'M'
      THEN PRHEMT();
      PRICHA(1, 5, SP);
      TWRT("M=EVENT :");
      PRIVAM(E);
      TWRT("#NL#");
    ELSE PRICHA(1, 5, SP);
      OUT(E . IDEN);
      TWRT("EVENT :");
      PRIVAM(E);
    END;
    E := E . NEXT;
  REP;
  PRICHA(2, 68, '*');
ENDPROC;
% READ A SET OF FAULTS %

EXT PROC(RAB, BYTE, INT, RAB, LABEL) GETSTR;
EXT PROC(RAB, RAB) LRAB;
EXT PROC(RAB, RAB) INT COMRAB;

INT PROC READSF ()
INT I := COFAULT;
L1: GETSTR(ST, "\x", I, "FAULT", L1);
WHILE COMRAB(ST, "***") = 0 DO
  LRAB(FAUL(I).L, ST);
  I := I + 1;
  FOR K := 1 TO LENGTH ST DO
    ST(K) := SP;
    REP;
L2: GETSTR(ST, "\x", I, "FAULT", L2);
REP;
COFAULT := I;
ENDPROC;
% READ TYPE OF THE UNITS %

EXT PROC(RAB, RAB) LRAB;
EXT PROC(RAB, RAB) INT COMRAB;
EXT PROC(RAB, BYTE, INT, RAB, LABEL) GETSTR;

ENT PROC READTU();
    INT I := COTYUN;
    L1: GETSTR(ST, '%', I, "UNIT", L1);
    WHILE COMRAB(ST, "***") = 0 DO
        LRAB(TYU(I) + L, ST);
        I := I+1;
        FOR K := 1 TO LENGTH ST DO
            ST(K) := SP;
        REP;
    L2: GETSTR(ST, '%', I, "UNIT", L2);
    REP;
    COTYUN := I;
ENDPROC;

PAGE
---
READTU.RTL
% READ FAULTS AND TRANSLATE INTO NUMBERS %

EXT PROC (RAB, BYTE, INT, RAB, LABEL) GETSTR;
EXT PROC (RAB, RAB) LRAB;
EXT PROC (RAB, RAB) INT COMRAb;

ENT PROC REFATR(RAB AUX) INT;
  IF COMRAb(AUX, "++") = 1 THEN
    L1:GETSTR(ST, ",", 1, "FAULT(*% TO TERM• INPUT)"., L1);
    IF COMRAb(ST, ",.*") = 1 THEN RETURN(0); END;
    ELSE LRAB(ST, AUX);
  END;
  FOR I := 1 TO MAXNOFAU TO DO
    IF COMRAb(FAUC, L, ST) = 1 THEN RETURN(I); END;
  REP;
  TWRT("#NL#UNDEFINED FAULT (REFATR,RTL)");
  RETURN(0);
ENDPROC;
I READ TYPE OF UNIT AND TRANSLATE INTO A NUMBER

EXT PROC(RAB, BYTE, INT, RAB, LABEL) GETSTR
EXT PROC(RAB, RAB) INT COMRAB

EXT PROC RETYUN () INT
L1:GETSTR(ST, 'Z', 1, "TYPE OF UNIT(+++++ TO TERMINATE INPUT)", L1);
   IF COMRAB(ST, "++++") = 1 THEN RETURN(0) END;
FOR I := 1 TO MAXNOUNIT DO
   IF COMRAB(TYU(I), ST) = 1 THEN RETURN(I) END;
   REP;
   TWRT("*NL*UNDEFINED UNIT(RETYUN.RTL)");
   RETURN(0);
ENDPROC;
OPTION(1) BC, TR;
TITLE
READ THE UNITS;

LET COMERR = 5000;
LET SP = OCT 48;
LET LF = OCT 12;

EXT PROC() REUNIT;
EXT PROC() RSXINT;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOPI, RSXOPO;
EXT PROC(INT) INT RSXSWI, RSXSWO;
EXT PROC(REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC() BYTE ECHOIN, COMIP;

SVC DATA RRERR:
LABEL ERL;
INT ERR;
PROC(INT) ERP;
ENDDATA;

SVC DATA RRSIO:
PROC() BYTE IN;
PROC(BYTE) OUT;
ENDDATA;

SVC DATA RRERRX:
INT LINENO;
BYTE UEFLAG, ERLUN;
INT RSXDSW;
ENDDATA;

DATA FILDAT:
ARRAY(70) BYTE FILE := "#SP(70)#";
ENDDATA;

ENT PROC RRJOB();
ERRLUN := 3;
RSXINT();
IN := COMIP;
IF IN() # 'R' OR IN() # 'E' OR IN() # 'U' OR IN() # ' ' THEN ERP(COMERR);
END;

TREAD(FILE, "=");
RSXOPO(1, FILE);
RSXSWO(1);
FOR J := 1 TO LENGTH FILE DO FILE(J) := SP;
REP;

TREAD(FILE, "#LF#"㈠;
IN := ECHOIN;
RSXOPI(2, FILE);
RSXSWI(2);
REUNIT();
RSXCL(1);
RSXCL(2);
% READ AND ALLOCATE UNITS %

EXT PROC() REF UNITDE NEXTFU;
EXT PROC(REF INT, INT, INT, INT, RAB, LABEL) GETINT;
EXT PROC() INT RETYUN;
EXT PROC(INT, INT, BYTE) PRICHA;

EXT PROC REUNIT();
    REF UNITDE U := NILUNI;
    TWRT("#NL(3)#TOPOLOGY OF THE SYSTEM PART : 1");
    PRICHA(1, 32, '"' );
    TWRT("#NL(2)#DESCRIPTION OF THE UNITS");
    PRICHA(1, 25, '"' );
    TWRT("#NL(3)#");
L1: GETINT(COUNT, INT, MAXCOUNT, INT, "HOW MANY UNITS IN THE SYSTEM ", L1);
    TWRT("#NL(2)#");
    FOR I := 1 TO COUNT DO
        U := NEXTFU();
        L2: GETINT(U, UNUM, INT, 32767, INT, "NO OF THE UNIT", L2);
            U*TY := BYTE(RETYUN());
        TWRT("#NL(2)#");
    REP;
        U*NEXU := NILUNI;
ENDPROC;
523

% READ VALUES OF MEASURED VARIABLES %

EXT PROC (REF INT,INT,INT,INT,RAB,LABEL) GETINT;
EXT PROC (RAB,BYTE,INT,RAB,LABEL) GETSTR;
EXT PROC (INT,RAB) GETBYT;
EXT PROC (INT,BYTE) PRICHA;

EXT PROC REVAME ();
REF VARIAB VI := NILVAR;
REF DATAV DI := NILDAT;
BYTE ANS,NF,NT;
INT N,K;
GETBYT(ANS,1,"DO YOU WANT TO LIST THE VARIABLES(Y/N)");
IF ANS = 'Y' THEN K := DISPLAC;END;
IF K = 0 THEN RETURN;END;
TWRT("%NL(3)#READING VALUES OF MEASURED VARIABLES#CR#");
PRICHA(1,36,*");
TWRT("%NL(2)"");
LA:=GETSTR(ST,','2,"NAME OF VARIABLE(ADD %* TO TERM*)",LA);
NF := ST(1);
NT := ST(2);
WHILE NF = 'Y' DO
   L1:=GETINT(N1,1,MAXNOVAR,1,"NO. OF VARIABLE",L1);
      FOR I := 1 TO MAXNOVAR DO
         IF VAB(I)*NOVA = #
         THEN GOTO L2;
         ELSEIF VAB(I)*NAMEF = NF AND
               VAB(I)*NAMEF = NT
         THEN VI := VAB(I);
         ELSE GOTO L2;
      END;
      IF VI*VALUES =: NILDAT AND VI*MESU = 'Y'
         THEN GOTO L3;
      ELSE TWRT("%NL#VARIABLE NOT MEASURED#CR#");
         GOTO EX;
      END;
   END;
   L2:=REP;
   TWRT("%NL#NO SUCH VARIABLE IN THE SYSTEM#CR#");
   GOTO EX;
   L3:=VI := VI*VALUES;
   GETBYT(ANS,1,"DO YOU WANT TO CHANGE THE LIMITS(Y/N)");
   IF ANS = 'Y'
      THEN
         L4:=GETINT(DI*HL,2,32767,1,"HIGH",L4);
         L5:=GETINT(DI*LL,2,32767,1,"LOW",L5);
         GETBYT(ANS,1,"DO YOU WANT TO CHANGE THE PRIORITY(Y/N)");
         IF ANS = 'Y'
            THEN
               L6:=GETINT(K,2,255,1,"PRIORITY",L6);
               DI*PRIO := BYTE(K);
            END;
      END;
   END;
   DI*PRVA := DI*ACTVA;
   L7:=GETINT(DI*ACTVA,2,32767,1,"VALUE OF VARIABLE",L7);
   EX:=GETSTR(ST,','2,"NAME OF VARIABLE(ADD %* TO TERM*)",EX);
   NF := ST(1);
   NT := ST(2);
   REP;
ENDPROC;
DATA OF A 'R' EVENT

INT PROC REVT(REF EVENT E);
E.BACKG := GATEP0;
E.NEXTG := GMJ;
ENDPROC;
OPTION(1) BC TRI
TITLE
READ SET OF FAULTS;

LET COMERR = 5088;
LET SP = OCT 48;
LET LF = OCT 12;

EXT PROC() READSF;
EXT PROC() RSXINTJ;
EXT PROC(INT) RSXCL;
EXT PROC(INT, REF ARRAY BYTE) RSXOPI, RSXPO;
EXT PROC(INT) INT RSXSWI, RSXSWO;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE ECHOIN, COMIP;

SVC DATA RRERR:
  LABEL ERLO;
  INT ERR;
  PROC(INT) ERP;
ENDDATA;

SVC DATA RRSIO:
  PROC () BYTE IN;
  PROC (BYTE) OUT;
ENDDATA;

SVC DATA RRERRX:
  INT LINEDO;
  BYTE UERFLG, ERRLUN;
  INT RSXDSW;
ENDDATA;

DATA FILDAT;
  ARRAY (78) BYTE FILE := "#SP(78)#";
ENDDATA;

ENT PROC RRRE()o;
  ERRLUN := 3;
  RSXINT();
  IN := COMIP;
  IF IN() != 'R' OR IN() != 'F' OR IN() != 'A' OR IN() != '
    THEN ERP(COMERR);
END;

TREAD(FILE, "=");
RSXOPI(1, FILE);
RSXSWO(1);

FOR J := 1 TO LENGTH FILE DO
  FILE(J) := SP;
REP;

TREAD(FILE, "#LF");
IN := ECHOIN;
RSXOPI(2, FILE);
RSXSWI(2);
READSF();
RSXCL(1);
RSXCL(2);
% TEST RANGE OF INTEGER %

ENT PROC RNGEPR (INT HIORLO);
TWRT("$NL,LF#OUT OF RANGE--");
IF HIORLO = -1
THEN TWRT("LO");
ELSE TWRT("HI");
END;
TWRT(" LIMIT = ");
RETURN;
ENDPROC;
OPTION(1) BC, TR;
TITLE
READ TYPE OF UNITS;

LET COMERR = 5000;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC() READTU;
EXT PROC() RXSINT;
EXT PROC(INT) RXSCL;
EXT PROC(INT, REF ARRAY BYTE) RXS0PI, RXS0PO;
EXT PROC(INT) INT RXSWS1, RXSWSO;
EXT PROC (REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC () BYTE ECHOIN, COMIP;

SVC DATA RRERR;
LABEL ERL;
INT ERN;
PROC(INT) ERP;
ENDDATA;

SVC DATA RSS10;
PROC () BYTE IN;
PROC (BYTE) OUT;
ENDDATA;

SVC DATA RRERRX;
INT LينENO;
BYTE UEFLAG, ERLUN;
INT RSXDSW;
ENDDATA;

DATA F1;
ARRAY(70) BYTE FILE := "#SF(70)#";
ENDDATA;

ENT PROC RRJOB();
ERLUN := 3;
RXSINT();
IN := COMIP;
IF IN() # 'R' OR IN() # 'T' OR IN() # 'U' OR IN() # '
 THEN ERF(COMERR);
END;
TREAD(FILE, "=");
RXS0PO(1, FILE);
RXSWSO(1);
FOR J := 1 TO LENGTH FILE DO
 FILE(J) := SP;
REP;
TREAD(FILE, "#LF#")
IN := ECHOIN;
RXS0PI(2, FILE);
RXSWSI(2);
READTU();
RXSCL(1);
RXSCL(2);
OPTION(1) BC, TR

TITLE
READ VALUES OF VARIABLES:

LET COMERR = 5000;
LET SP = OCT 40;
LET LF = OCT 12;

EXT PROC(REVAME);  EXT PROC(RSXINT);
EXT PROC(RSXCL);
EXT PROC(RSXOPI, RSXOPO);  EXT PROC(INT, INT) RSXSWI, RSXSWO;
EXT PROC(REF ARRAY BYTE, REF ARRAY BYTE) INT TREAD;
EXT PROC() BYTE COMIP;

SVC DATA RRERR;
LABEL ERLL;
INT ERNL;
PROC(INT) ERP;
ENDDATA;

SVC DATA RRSIO;
PROC() BYTE IN;
PROC(BYTE) OUT;
ENDDATA;

SVC DATA RRERRX;
INT LINENO;
BYTE UEFLAG, ERRLUN;
INT RSXDSW;
ENDDATA;

DATA FILDAT;
ARRAY (35) BYTE IFILE := "#SP(35)#";
ARRAY (35) BYTE OFILE := "#SP(35)#";
ENDDATA;

ENT PROC RRJOB();
ERRLUN := 3;
IN := COMIP;
IF IN(*) # 'R' OR IN(*) # 'V' OR IN(*) # 'A' OR IN(*) # '

THEN ERP(COMERR);
END;

TREAD(OFILE, "=");
TREAD(IFILE, "#LF#");
RSXINT();
RSXOPO(1, OFILE);
RSXSWO(1);
RSXOPI(2, IFILE);
RSXSWI(2);
REVAME();
RSXCL(1);
RSXCL(2);
ENDPROC;
% SET BOUND. COND. & N.A.C. FOR MINITREE%

EXT PROC (RAH) INT REFATR;
EXT PROC (REF INT, INT) SETBIT;

INT PROC SBC (REF BOUND, CO BOCO);

INT J;
INT FAEV := BOCO * VARSC * FAULT;
IF FAEV = 1 THEN SETBIT (BOCO * NA1, 2);
ELSEIF FAEV = 2 THEN SETBIT (BOCO * NA1, 1);
ELSEIF FAEV = 3 THEN SETBIT (BOCO * NA1, 4);
ELSEIF FAEV = 4 THEN SETBIT (BOCO * NA1, 3);

% MORE FAULTS CAN BE ADDED %

END;
TWRT("#NL# NOT ALLOWED CONDITIONS : #NL#")
J := REFATR("++");
WHILE J # 0 DO
IF J <= 16 THEN SETBIT (BOCO * NA1, J);
ELSEIF J <= 32 THEN SETBIT (BOCO * NA2, J - 16);
ELSEIF J <= 48 THEN SETBIT (BOCO * NA3, J - 32);

END;
J := REFATR("++");
REP;
ENDPROC;
% SET A BIT IN A BYTE %

ENT PROC SBITBY(REF BYTE BYT, INT BITNUM);
  VAL BYT := BYTE ( BYT LOR (128 SRL (8-BITNUM)));
ENDPROC;
% SCAN THE VARIABLES %

EXT PROC (INT) LORIMI;
EXT PROC (RAB) INT REFACTR;
EXT PROC (REF VARIAB) INT CHEVAC;
EXT PROC (BYTE, INT) INT GETBIB;

INT PROC SCANVA();
  INT I ;
  REF VARIAB V := NILVAR; VP := NILVAR;
  WHILE Ø = Ø DO
    I := I ;
    V := VAB(I);
    WHILE V.NOVA ≠ Ø DO
      IF V.MESU = 'Y' AND CHEVAC(V) = 1
        THEN IF GETBIB(V.EX, 2) = Ø
              THEN IF VP := NILVAR
                  THEN VP := V;
          ELSEIF VP.VALUES.PRIOR = V.VALUES.PRIOR
              THEN VP := V;
        END;
      END;
      I := I + 1;
      V := VAB(I);
    END;
    IF VP := NILVAR THEN RETURN; END;
    VST := VP;
  BLOCK
    REF DATAU VI := VP.VALUES;
    IF VI.ACTVA > VI.HL
      THEN LORIMI(REFACTR("HI"));
    ELSEIF VI.ACTVA < VI.LL
      THEN LORIMI(REFACTR("LO"));
  END;
  ENDBLOCK;
  VP := NILVAR;
  BT := Ø;
  C := '+';
  FLAG := Ø;
  REP;
ENDPROC;
% SET BOUND. COND. AND N. A. C. IN TREE %

EXT PROC (INT, REF INT) COPYB1;
EXT PROC () REF BOUNDECO NEXTFB;

ENT PROC SETBC ();
    REF BOUNDECO B1 := EST.NEXTG.BC;
    GST.BC := NEXTFB();
    BCST := GST.BC;
    BCST.VARBC := EAST;
    COPYB1(B1.NA1, BCST.NA1);
    COPYB1(B1.NA2, BCST.NA2);
    COPYB1(B1.NA3, BCST.NA3);
ENDPROC;
% SET A BIT IN AN INTEGER %

ENT PROC SETBIT(REF INT WORD, INT BITNUM);

   VAL WORD := WORD LOR (1 SLL (BITNUM - 1));
ENDPROC;

----------

535
% SET A GATE IN TREE %

EXT PROC () REF GATE NEXTFG;
EXT PROC () SETBC;

ENT PROC SETGA();
  EAST$NEXTG := NEXTFG();
  GST := EAST$NEXTG;
  GST$CLASS := GST$CLASS OR EST$NEXTG$CLASS;
  GST$BACKE := EAST;
  SETBC();
ENDPROC;
EXT PROC () REF STREAM NEXTFS;

ENT PROC SETSTR(REF STREAM S) REF STREAM;
  WHILE S·NEXTS #: NILSTM DO
    S·NEXTS := S·NEXTS;
  REP;
  S·NEXTS := NEXTFS();
  S := S·NEXTS;
  RETURN(S);
ENDPROC;
% SET GATE FOR MINITREES %

EXT PROC (REF INT, INT, INT, INT, RAB, LABEL) GETINTJ
EXT PROC () REF BOUNDECO NEXTFB;
EXT PROC () REF GATE NEXTFGJ;
EXT PROC (REF BOUNDECO) SBC;
EXT PROC (REF BYTE, INT) SHITBY;

EXT PROC SGA(REF EVENT E);
  INT C;
  E.NEXTG := NEXTFG();
  GATEPO := E.NEXTG;
  LI := GETINT(C, 0, 2, 1, "TYPE OF GATE(0-OR, 1-AND, 2-EXOR)", LI);
  SHITBY(GATEPO, CLASS, C);
  GATEPO.BACKE := E;
  GATEPO.NEXT := E.NEXT;
  GATEPO.BC := NEXTFB();
  GATEPO.BC.VARBC := E;
  SBC(GATEPO.BC);
ENDPROC;
OPTION(1) BC, TR;
TITLE

STDo

**REMARKS OF THE DATA BASE**

**DESCRIPTION OF THE RECORDS**

**MODE EVENT**

BYTE IDEN, UNID, VAR, INOU,
REF VARIAB NAVA,
INT FAULT,
REF EVENT NEXT, TOP,
REF GATE BACKG, NEXTG,
REF UNITDE UNNO,
FRAC PROB);

**MODE GATE**

BYTE CLASS, EXTRA,
REF EVENT BACKG, NEXTG,
REF GATE NEGATE,
REF BOUNDECO BC);

**MODE BOUNDECO**

REF EVENT VARGBC,
INT NA1, NA2, NA3,
REF BOUNDECO NEXTBC);

**MODE INDEX**

BYTE TYPE, TEX,
REF EVENT FIRSEL);

**MODE UNITDE**

INT UNUM,
BYTE TY, TYE,
REF STREAM INST, OUTST,
REF UNITDE NEXU);
MODE STREAM (BYTE SIO, EIO,
               REF UNITDE FROMU, TOUN,
               REF VARIAB NAVAR,
               REF STREAM NEXTS);

MODE VARIAB (BYTE NAMEF, NAMET, MESU, EX,
               INT NOVA,
               REF STREAM BACKS,
               REF VARIAB NEXVAR,
               REF DATAV VALUES);

MODE DATAV (INT HL, LL, FRVA, ACTVA,
            BYTE FRIO, PREX);

MODE FAS (ARAB L);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

EXT DATA MINIT:

% ARRAY OF EVENTS %
ARRAY (MAXNOE) EVENT EVT;

% ARRAY OF GATES %
ARRAY (MAXNOG) GATE GAT;

% ARRAY OF BOUND. CONDITIONS %
ARRAY (MAXNOG) BOUNDECO BCO;

% ARRAY TO LOCATE TOP EVENTS OF UNITS %
ARRAY (MAXNOUNIT) INDEX LOT;

% ARRAY OF FAULTS %
ARRAY (MAXNOFAULT) FAS FAU;

% ARRAY OF TYPES OF UNITS %
ARRAY (MAXNOUNIT) FAS T'U;

% NIL DECLARATIONS %
EVENT NILLEV;
EVENT NILNEX;
GATE NILGA;
GATE NILNEG;
GATE QM;
BOUNDECO NILBC;
BOUNDECO NILNEBC;
% INITIATION OF VARIABLES %

REF EVENT FREEV;
REF GATE FREEGA;
REF GATE GATEPO;
REF BOUNDECO FREEBC;
INT CI;
INT CF;
INT COLO;
INT ITE;
INT GONFAULT;
INT COTYUN;
ARRAY (ARESIZ) BYTE ST;
ENDDATA;

PEND------------------------------

% EXT DATA TOPOLOGY; %

% ARRAYS USED TO DESCRIBE THE TOPOLOGY OF THE SYSTEM %

% ARRAY OF UNITS %
ARRAY (MAXNOUNIT) UNITDE UNI;

% ARRAY OF STREAMS %
ARRAY (MAXNOSTR) STREAM STR;

% ARRAY OF VARIABLES %
ARRAY (MAXNOVAR) VARIAB VAB;

% ARRAY OF MEASURED VARIABLES %
ARRAY (MAXNOMEVAR) DATAV DAT;

% NIL DECLARATIONS %
UNITDE NILUNI;
UNITDE NILUNE;
STREAM Nilstm;
STREAM Nilstn;
STREAM Nilsns;
STREAM NILOUT;
VARIAB NILVAR;
VARIAB NILVAN;
DATAV NILDAT;

% INITIATION OF VARIABLES %
REF UNITDE FREEUN;
REF STREAM FREEST;
REF VARIAB FREEVA;
INT COUNTV;
INT COUNIT;
BYTE C;

ENDDATA;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

EXT DATA TREES;

% INITIATION OF VARIABLES %

INT RTD;
INT ET;
INT FAST;
INT FLAG;
REF EVENT EAST;
REF EVENT EST;
REF GATE GST;
REF BOUNDECO BGST;
REF UNITDE UST;
REF STREAM SST;
REF VARIAB VST;

ENDDATA;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SVC DATA RRSCI;
    PROC() BYTE IN;
    PROC (BYTE) OUT;
ENDDATA;

SVC DATA RRERRX;
    INT LINENO;
    BYTE UEFLAG,ERRLUN;
    INT RSXDSW;
ENDDATA;

EXT PROC(RAB) TWRT;
EXT PROC(INT) IWRIT;
EXT PROC() INT IREAD;
EXT PROC(RAB, RAB) INT TREAD;
% TEST DIRECTIVE STATUS WORD %

EXT PROC (INT) RRGEL;

ENT PROC TESDSW (INT E);
    IF RSXDSW < 0
    THEN RRGEL(E);
    END;
    RETURN;
ENDPROC;

% TEST THE RANGE OF A FRACTION

EXT PROC (FRAC, INT, LABEL) RANGEF;

ENT PROC TESTF(FRAC I, FRAC LO, LABEL ERRLAB):
    IF I < LO
        THEN RANGEF(LO, -1, ERRLAB);
    END;
ENDPROC;

PAGE TESTF .RTL
% TEST THE RANGE OF THE INTEGER %

EXT PROC (INT,INT,LABEL) RANGEP;

ENT PROC TESTI (INT I,LO,HI,LABEL ERRLAB);
  IF I < LO
  THEN RANGEP(LO,-1,ERRLAB);
  ELSEIF I > HI
  THEN RANGEP(HI,+1,ERRLAB);
  END;
  RETURN;
ENDPROC;

ENDPROCI
% TEST SPACE %

EXT PROC (INT) NCLS;

ENT PROC TESTSP(RAB A, INT LFEED, REF BYTE B);
    NCLS(LFEED);
    TWRT(A);
    IF B = SP
        THEN OUT('S');
    ELSE OUT(B);
END;
ENDPROC;
% DATA OF A 'T' EVENT %

ENT PROC TEVT (REF EVENT E);
   E·BACKG := GATEPO;
   E·NEXTG := QM;
ENDPROC;
EXT PROC () REF EVENT.NEXTFE;
EXT PROC () SETGA;
EXT PROC (REF BYTE, INT) SBITBY;

EXT PROC TOEVT ();
EAST := NEXTFE();
EAST.IDEN := EST.IDEN;
EAST.UNID := EST.UNID;
EAST.VAR := EST.VAR;
EAST.INOU := EST.INOU;
EAST.NAVA := VST;
SBITBY(VST.EX,2);
EAST_FAULT := EST_FAULT;
EAST.NEXT := NIL EV;
SETGA();
EAST.UNNO := UST;
EAST.PROB := EST.PROB;
BT := 1;
ENDPROC;
ENT PROC TRNOWR( INT 1);
  IF I ≠ 0 THEN
    TWRT(FAU(I)*L);
  ELSE IWRT(I);
  END;
ENDPROC
% PRINT ARRAY BYTE WITHOUT SPACES %

ENT PROC TWRAB(RAB A);
  FOR I := 1 TO LENGTH A DO
    IF A(I) = SF THEN RETURN;
    ELSE OUT(A(I));
    END;
  REP;
ENDPROC;

PAGE
------
TWRAB .RTL
% GET THE VALUES OF MEASURED VARIABLES %

EXT PROC (REF BYTE, INT, RAB) GETBYTE;
EXT PROC (REF INT, INT, INT, RAB, LABEL) GETINT;

ENT PROC VALUE (REF VARIAB V);
REF DATAV VA := NILDAT;
INT K;
IF COUNTV <= MAXNOVEMVAR
THEN VA := DAT (COUNTV);
   L1 := GETINT (VA, 0, 32767, 1, "HIGH", L1);
   L2 := GETINT (VA, 0, 32767, 1, "LOW", L2);
   L3 := GETINT (K, 0, 255, 1, "PRIORITY OF THE VAR.", L3);
   VA.PRI0 := BYTE (K);
   VALUES := VA;
   COUNTV := COUNTV + 1;
ELSE TWRT("#NL#NO MORE ELEMENTS IN DAT(K)";
"AVAILABLE VALUE.RTL")
END;
ENDPROC;
### Table III.1

**Look-up Table for Faults**

<table>
<thead>
<tr>
<th>Number Used</th>
<th>Name of Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HI</td>
</tr>
<tr>
<td>2</td>
<td>LO</td>
</tr>
<tr>
<td>3</td>
<td>OPEN</td>
</tr>
<tr>
<td>4</td>
<td>CLOSED</td>
</tr>
<tr>
<td>5</td>
<td>FL-EX-ENV</td>
</tr>
<tr>
<td>6</td>
<td>BLOCKAGE</td>
</tr>
<tr>
<td>7</td>
<td>LK-IP-ENV</td>
</tr>
<tr>
<td>8</td>
<td>LK-HP-ENV</td>
</tr>
<tr>
<td>9</td>
<td>WIDE-OPEN</td>
</tr>
<tr>
<td>10</td>
<td>SHUT</td>
</tr>
<tr>
<td>11</td>
<td>BLOC-OUTL</td>
</tr>
<tr>
<td>12</td>
<td>BLOC-INLE</td>
</tr>
<tr>
<td>13</td>
<td>FAI-TO-OP</td>
</tr>
<tr>
<td>14</td>
<td>MANUAL</td>
</tr>
<tr>
<td>15</td>
<td>CONT-STCK</td>
</tr>
<tr>
<td>16</td>
<td>SENS-STCK</td>
</tr>
<tr>
<td>17</td>
<td>VALV-STCK</td>
</tr>
<tr>
<td>18</td>
<td>FAI-TO-CL</td>
</tr>
<tr>
<td>19</td>
<td>SET-PO-HI</td>
</tr>
<tr>
<td>20</td>
<td>FAIL-OPEN</td>
</tr>
<tr>
<td>21</td>
<td>FAIL-HI</td>
</tr>
<tr>
<td>22</td>
<td>SEN-FA-LO</td>
</tr>
<tr>
<td>23</td>
<td>SET-PO-LO</td>
</tr>
<tr>
<td>24</td>
<td>FAI-CLOSE</td>
</tr>
<tr>
<td>25</td>
<td>FAIL-LO</td>
</tr>
<tr>
<td>26</td>
<td>SEN-FA-HI</td>
</tr>
<tr>
<td>27</td>
<td>CONT-F-HI</td>
</tr>
<tr>
<td>28</td>
<td>CONT-F-LO</td>
</tr>
<tr>
<td>29</td>
<td>NO-CHANGE</td>
</tr>
<tr>
<td>30</td>
<td>EXT-FIRE</td>
</tr>
<tr>
<td>31</td>
<td>MECH-FAIL</td>
</tr>
<tr>
<td>32</td>
<td>OTHER-CAU</td>
</tr>
<tr>
<td>33</td>
<td>Z-HI</td>
</tr>
<tr>
<td>34</td>
<td>Z-LO</td>
</tr>
<tr>
<td>35</td>
<td>A</td>
</tr>
<tr>
<td>36</td>
<td>B</td>
</tr>
<tr>
<td>37</td>
<td>C</td>
</tr>
<tr>
<td>38</td>
<td>D</td>
</tr>
<tr>
<td>39</td>
<td>DUMMY</td>
</tr>
<tr>
<td>40</td>
<td>NO-FLOW</td>
</tr>
<tr>
<td>41</td>
<td>SI-STR-PL</td>
</tr>
<tr>
<td>42</td>
<td>NO-SIGNAL</td>
</tr>
<tr>
<td>43</td>
<td>GY3</td>
</tr>
<tr>
<td>44</td>
<td>SHUTDOWN</td>
</tr>
<tr>
<td>45</td>
<td>COMP-BLOC</td>
</tr>
</tbody>
</table>
### Table III.2

**Look-up Table for Type of Units**

<table>
<thead>
<tr>
<th>Number Used</th>
<th>Type of Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CENT-PUMP</td>
</tr>
<tr>
<td>2</td>
<td>CLOSED-TK</td>
</tr>
<tr>
<td>3</td>
<td>CNTRLV-SC</td>
</tr>
<tr>
<td>4</td>
<td>CNTRL-VAL</td>
</tr>
<tr>
<td>5</td>
<td>CNTROLLER</td>
</tr>
<tr>
<td>6</td>
<td>DUMMY-H</td>
</tr>
<tr>
<td>7</td>
<td>DUMMY-T</td>
</tr>
<tr>
<td>8</td>
<td>HEAT-EX</td>
</tr>
<tr>
<td>9</td>
<td>PIPE</td>
</tr>
<tr>
<td>10</td>
<td>SENSOR-Q</td>
</tr>
<tr>
<td>11</td>
<td>SENSOR-P</td>
</tr>
<tr>
<td>12</td>
<td>SENSOR-T</td>
</tr>
<tr>
<td>13</td>
<td>VALVE</td>
</tr>
</tbody>
</table>
### Table III.3

**MODE BOUNDECO Description**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARBC</td>
<td>REF EVENT</td>
<td>Pointer to a specific event</td>
</tr>
<tr>
<td>NA1(*)</td>
<td>INT</td>
<td>Integer that stores 16 different faults</td>
</tr>
<tr>
<td>NA2(*)</td>
<td>INT</td>
<td>Integer that stores 16 different faults</td>
</tr>
<tr>
<td>NA3(*)</td>
<td>INT</td>
<td>Integer that stores 16 different faults</td>
</tr>
<tr>
<td>NEXTBC</td>
<td>REF BOUNDECO</td>
<td>Pointer to next Boundary Condition</td>
</tr>
</tbody>
</table>

### Table III.4

**MODE DATAV Description**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL</td>
<td>INT</td>
<td>Stores the High limit of the variable</td>
</tr>
<tr>
<td>LL</td>
<td>INT</td>
<td>Stores the Low limit of the variable</td>
</tr>
<tr>
<td>PRVA</td>
<td>INT</td>
<td>Stores the previous value of the variable</td>
</tr>
<tr>
<td>ACTVA</td>
<td>INT</td>
<td>Stores the actual value of the variable</td>
</tr>
<tr>
<td>Prio</td>
<td>BYTE</td>
<td>Stores the priority of the variable</td>
</tr>
<tr>
<td>PREX</td>
<td>BYTE</td>
<td>Spare field</td>
</tr>
</tbody>
</table>
Table III.5

**MODE EVENT Description**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEN</td>
<td>BYTE</td>
<td>Describes the type of event according to the definitions of Chapter 3</td>
</tr>
<tr>
<td>UNID</td>
<td>BYTE</td>
<td>Describes the name of the unit to which the event is related</td>
</tr>
<tr>
<td>VAR</td>
<td>BYTE</td>
<td>Describes the variable of the event.</td>
</tr>
<tr>
<td>INOU</td>
<td>BYTE</td>
<td>Describes the type of variable (output, input, internal or no variable)</td>
</tr>
<tr>
<td>NAVA</td>
<td>REF VARIAB</td>
<td>Pointer to the variable related to the event in the topology. (Only used in the construction of the fault trees)</td>
</tr>
<tr>
<td>FAULT</td>
<td>INT</td>
<td>Describes the name of the fault</td>
</tr>
<tr>
<td>NEXT</td>
<td>REF EVENT</td>
<td>Pointer to next event (if there are any)</td>
</tr>
<tr>
<td>TOP</td>
<td>REF EVENT</td>
<td>Pointer to next top event used only in the construction of minitrees</td>
</tr>
<tr>
<td>BACKG</td>
<td>REF GATE</td>
<td>Pointer to the backgate of the event</td>
</tr>
<tr>
<td>NEXTG</td>
<td>REF GATE</td>
<td>Pointer to the next gate of the event. (If there are any)</td>
</tr>
<tr>
<td>UNNO</td>
<td>REF UNITDE</td>
<td>Pointer to the unit to which this event belongs in the topology. (Only used in the construction of the fault trees)</td>
</tr>
<tr>
<td>PROB</td>
<td>FRAC</td>
<td>Stores the probability of the event. (Basic events only)</td>
</tr>
</tbody>
</table>
### Table III.6

**MODE GATE Description**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS(*)</td>
<td>BYTE</td>
<td>Describes the type of gate (AND, OR, EX-OR).</td>
</tr>
<tr>
<td>EXTRA</td>
<td>BYTE</td>
<td>Spare field.</td>
</tr>
<tr>
<td>BACKE</td>
<td>REF EVENT</td>
<td>Pointer to the back event of the gate</td>
</tr>
<tr>
<td>NEXTE</td>
<td>REF EVENT</td>
<td>Pointer to the first next event of the gate</td>
</tr>
<tr>
<td>NEGATE</td>
<td>REF GATE</td>
<td>Pointer to the next gate</td>
</tr>
<tr>
<td>BC</td>
<td>REF BOUNDEC</td>
<td>Pointer to the Boundary Conditions related to the back event of the gate</td>
</tr>
</tbody>
</table>

### Table III.7

**MODE INDEX Description**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>BYTE</td>
<td>Describes the type of unit.</td>
</tr>
<tr>
<td>TEX</td>
<td>BYTE</td>
<td>Spare field</td>
</tr>
<tr>
<td>FIRSEL</td>
<td>REF EVENT</td>
<td>Pointer to the first event in the set of minitrees for the unit described by the field TYPE</td>
</tr>
</tbody>
</table>
### Table III.8

#### MODE STREAM Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIO(*)</td>
<td>BYTE</td>
<td>Describes the type of stream (input or output).</td>
</tr>
<tr>
<td>EIO</td>
<td>BYTE</td>
<td>Spare field</td>
</tr>
<tr>
<td>FROMU</td>
<td>REF UNITDE</td>
<td>Pointer to the unit where the stream comes from.</td>
</tr>
<tr>
<td>TOUN</td>
<td>REF UNITDE</td>
<td>Pointer to the unit where the stream goes to</td>
</tr>
<tr>
<td>NAVAR</td>
<td>REF VARIAB</td>
<td>Pointer to the first variable of the stream</td>
</tr>
<tr>
<td>NEXU</td>
<td>REF UNITDE</td>
<td>Pointer to the next stream</td>
</tr>
</tbody>
</table>

### Table III.9

#### MODE UNITDE Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNUM</td>
<td>INT</td>
<td>Describes the No. of the unit according to the No. given in the topology of the system</td>
</tr>
<tr>
<td>TY</td>
<td>BYT</td>
<td>Describes the type of unit</td>
</tr>
<tr>
<td>TYE</td>
<td>BYTE</td>
<td>Spare field</td>
</tr>
<tr>
<td>INST</td>
<td>REF STREAM</td>
<td>Pointer to the first input stream of the unit</td>
</tr>
<tr>
<td>OUTST</td>
<td>REF STREAM</td>
<td>Pointer to the first output stream of the unit</td>
</tr>
<tr>
<td>NEXU</td>
<td>REF UNITDE</td>
<td>Pointer to the next unit of the system</td>
</tr>
<tr>
<td>Field Name</td>
<td>Mode</td>
<td>Function</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>NAMEF</td>
<td>BYTE</td>
<td>Describes the first letter of the complete variable</td>
</tr>
<tr>
<td>NAMEF</td>
<td>BYTE</td>
<td>Describes the other letter of the complete variable. (No letter will be assigned if it refers to an internal variable)</td>
</tr>
<tr>
<td>MESU</td>
<td>BYTE</td>
<td>Is used to know if the variable is a measured one or not</td>
</tr>
<tr>
<td>EX</td>
<td>BYTE</td>
<td>Is used only during the definition of the topology</td>
</tr>
<tr>
<td>NOVA</td>
<td>INT</td>
<td>Describes the No. of the variable</td>
</tr>
<tr>
<td>BACKS</td>
<td>REF STREAM</td>
<td>Pointer to the back stream, (the stream which carries the variable in the topology)</td>
</tr>
<tr>
<td>NEXVAR</td>
<td>REF VARIAB</td>
<td>Pointer to next variable</td>
</tr>
<tr>
<td>VALUES</td>
<td>REF DATVAR</td>
<td>Pointer to the values of the variable (only if it is a measured one)</td>
</tr>
</tbody>
</table>
### Table III.11

**Description of the Pointers Defined in the Data Base**

<table>
<thead>
<tr>
<th>Name of Variable</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREEV</td>
<td>REF EVENT</td>
<td>Pointer to the next free event in the array of events</td>
</tr>
<tr>
<td>FREEGA</td>
<td>REF GATE</td>
<td>Pointer to the next free gate in the array of gates</td>
</tr>
<tr>
<td>GATEPO</td>
<td>REF GATE</td>
<td>Pointer to the actual gate being used when developing a fault tree</td>
</tr>
<tr>
<td>FREEBC</td>
<td>REF BOUNDECO</td>
<td>Pointer to the next free Boundary Condition in the array of Boundary Conditions</td>
</tr>
<tr>
<td>FREEUN</td>
<td>REF UNITDE</td>
<td>Pointer to the next free unit in the array of units</td>
</tr>
<tr>
<td>FREEST</td>
<td>REF STREAM</td>
<td>Pointer to the next free stream in the array of streams</td>
</tr>
<tr>
<td>FREEVA</td>
<td>REF VARIAB</td>
<td>Pointer to the next free variable in the array of variables</td>
</tr>
<tr>
<td>EAST</td>
<td>REF EVENT</td>
<td>Pointer to the actual events being used during the development of a fault tree</td>
</tr>
<tr>
<td>EST</td>
<td>REF EVENT</td>
<td>Pointer to the event of the minitree that is being used at each stage of the fault tree development</td>
</tr>
<tr>
<td>GST</td>
<td>REF GATE</td>
<td>Pointer to the gate of the minitree that is being used at each stage of the fault tree development</td>
</tr>
<tr>
<td>BCST</td>
<td>REF BOUNDECO</td>
<td>Pointer to the Boundary Condition of the minitree that is being used at each stage of the fault tree development</td>
</tr>
<tr>
<td>UST</td>
<td>REF UNITDE</td>
<td>Pointer to the unit of the topology being used at each stage of the development of a fault tree</td>
</tr>
<tr>
<td>SST</td>
<td>REF STREAM</td>
<td>Pointer to the stream of the topology that is being used at each stage of the fault tree development</td>
</tr>
<tr>
<td>VST</td>
<td>REF VARIAB</td>
<td>Pointer to the variable of the topology that is being used at each stage of the fault tree development</td>
</tr>
</tbody>
</table>
Table III.12
Description of the Tasks

<table>
<thead>
<tr>
<th>Name of the Task</th>
<th>Function</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMT</td>
<td>Build Minitrees</td>
<td>None</td>
</tr>
</tbody>
</table>
| BTR*             | Build Trees | Ø-Design  
|                  |           | 1-Real Time |
| CAL              | Link and clear the arrays related to the minitrees and trees (Events, gates, Boundary Conditions) | 0-Link only  
|                  |           | 1-Clear arrays  
|                  |           | 2-Clear and link arrays |
| DEB*             | Print the arrays of the Database | *1+9*  
|                  |           | 1+3  
|                  |           | A  
|                  |           | B  
|                  |           | A-No. of arrays  
|                  |           | B-Part of the array |
| DES              | Description of the units in the topology | None |
| LIT              | Link the arrays related to the topology | None |
| PMT              | Print the minitrees | None |
| PRI              | Print the topology | None |
| PTR              | Print the fault tree | None |
| REU              | Description of the streams and variables in the topology | None |
| RFA              | Read the set of faults | None |
| RTU              | Read the type of units | None |
| RVA              | Task to read the values of measured variables and modify priorities | None |

*Overlays are used in these tasks.*
OPTION(1) BC, TR;
TITLE
DATA BASE:

******************************************************************************
LET ARBSIZ = 10;
LET ARAB = ARRAY(ARBSIZ) BYTE;
LET RAH = REF ARRAY BYTE;
LET SP = OCT 40;

% RESTRICTIONS OF THE DATA BASE %
******************************************************************************
LET MAXNOE = 500;
LET MAXNOG = 499;
LET MAXNOGMI = 149;
LET MAXNOUNIT = 151;
LET MAXNOH UnitM1 = 14;
LET MAXNOSTR = 501;
LET MAXNO STRM1 = 491;
LET MAXNOVAR = 501;
LET MAXNOVARMI = 491;
LET MAXNOVARMI = 491;
LET MAXNOVAR = 501;
LET MAXNOVARMI = 491;
LET MAXNOVARMI = 491;
LET MAXNOVARS = 501;
LET MAXNOVARS = 501;
LET MAXNOVARS = 501;
LET MAXNOVARS = 501;

% DESCRIPTION OF THE RECORDS %
******************************************************************************

MODE EVENT (BYTE IDEN, UNID, VAR, INOU,
REF VARIAB NAVA,
INT FAULT,
REF EVENT NEXT, TOP,
REF GATE BACK, NEXTG,
REF UNITDE UNNO,
FRAG PROB);

MODE GATE (BYTE CLASS, EXTRA,
REF EVENT BACK, NEXTG,
REF GATE NEGATE,
REF BOUNDECO BC);

MODE BOUNDECO (REF EVENT VARBC,
INT NA1, NA2, NA3,
REF BOUNDECO NEXTBC);

MODE INDEX (BYTE TYPE, TEX,
REF EVENT FIRSEL);

MODE UNITDE (INT UNNUM,
BYTE TY, TYE,
REF STREAM INST, OUTST,
REF UNITDE NEXU);

MODE STREAM (BYTE SI0, EI0,
REF UNITDE FROMU, TOUN,
REF VARIAB NAVAR,
REF STREAM NEXTS);

Fig. III.2.3.1 Data Base Program : /continued
MODE VARIAB (BYTE NAMEF,NAMET, MESSU, EX,
INT NOVA,
REF STREAM BACKS,
REF VARIAB NEXVAR,
REF DATAV VALUES);

MODE DATAV (INT HL, LL, PRVA, ACTVA,
BYTE PRIQ, PREX);

MODE FAS (ARAB L);

******************************************************************************

ENT DATA MINIT;

% ARRAY OF EVENTS %

ARRAY (MAXNOE) EVENT EVT :=
((SP,SP,SP,0,NILVAR,0,NILEV,NILEV,
  NILGA,NILGA,NILUNI,0,2B0) (MAXNOEM1),
 (SP,SP,SP,0,NILVAR,0,NILEX,NILEV,
  NILGA,NILGA,NILUNI,0,2B0));

% ARRAY OF GATES %

ARRAY (MAXNOG) GATE GAT1 :=
((0,SP,NILEV,NILEV,NILGA,NILBC) (MAXNOGM1),
 (0,SP,NILEV,NILEV,NILNEG,NILBC));

% ARRAY OF BOUND CONDITIONS %

ARRAY (MAXNOG) BOUNDECO BCO :=
((NILEV,0,0,2,NILBC) (MAXNOGM1),
 (NILEV,0,2,0,NILNEBC));

% ARRAY TO LOCATE TOP EVENTS OF UNITS %

ARRAY (MAXNOUNIT) INDEX LOT :=
((SP,SP,NILEV) (MAXNOUNIT));

% ARRAY OF FAULTS %

ARRAY (MAXNOFAULT) FAS FAU :=
(("#SP(ARBSIZ)#") (MAXNOFAULT));

% ARRAY OF TYPES OF UNITS %

ARRAY (MAXNOUNIT) FAS TYU :=
(("#SP(ARBSIZ)#") (MAXNOUNIT));

% NIL DECLARATIONS %

EVENT NILEV :=
(SP,SP,SP,SP,NILVAR,0,NILEV,NILEV,NILGA,NILGA,NILUNI,
  0,2B0);

EVENT NILNEX :=
("**","**","**","**","**",NILEV,0,NILEV,NILEV,NILNEG,NILNEG,
  NILUNI,0,2B0);

GATE NILGA :=

Fig. III.2.3.1 /continued
GATE NILNEG := ('*','*','NILNEX','NILNEX','NILGA','NILNEBC');

GATE QM := ('?','SP','NILEV','NILEV','NILGA','NILBC');

BOUNDECO NILBC := (NILEV, 0, 0, 0, NILBC);

BOUNDECO NILNEBC := (NILNEX, 0, 0, 0, NILBC);

% INITIATION OF VARIABLES %

REF EVENT FREEV := EVT(1);
REF GATE FREEGA := GAT(1);
REF GATE GATEPO := GAT(1);
REF BOUNDECO FREEBC := BCO(1);
INT CI := 1;
INT CF := 1;
INT COLO := 1;
INT ITE := 1;
INT COFAULT := 1;
INT COTYUN := 1;
ARRAY (ARBSIZ) BYTE ST := (SP(ARBSIZ));

ENDDATA;

%******************************************************************************%

ENT DATA TOPOLOGY;

% ARRAYS USED TO DESCRIBE THE TOPOLOGY OF THE SYSTEM %

% ARRAY OF UNITS %

ARRAY (MAXNOUNIT) UNITDE UNI :=
((0, SP, SP, NILSTM, NILSTM, NILUNI) (MAXNOUNIT)),
((0, SP, SP, NILSTM, NILSTM, NILUNE))

% ARRAY OF STREAMS %

ARRAY (MAXNOSTR) STREAM STR :=
((0, 0, NILUNI, NILUNI, NILVAR, NILSTM) (MAXNOSTR)),
((0, 0, NILUNI, NILUNI, NILVAR, NILSTM))

% ARRAY OF VARIABLES %

ARRAY (MAXNOVAR) VARIAB VAB :=
((SP, SP, SP, 0, 0, NILSTM, NILVAR, NILDAT) (MAXNOVAR)),
((SP, SP, SP, 0, 0, NILSTM, NILVAR, NILDAT))

% ARRAY OF MEASURED VARIABLES %

ARRAY (MAXNOMEVAR) DATAV DAT :=
((0, 0, 0, 0, SP, SP) (MAXNOMEVAR))

% NIL DECLARATIONS %

Fig. III.2.3.1 /continued
UNITDE NILUNI :=
(0, SP, SP, NILSTM, NILSTM, NILUNI);

UNITDE NILUNE :=
(0, '*', '*', '*', NILSTM, NILSTM, NILUNI);

STREAM NILSTM :=
(SP, SP, NILUNI, NILUNI, NILVAR, NILSTM);

STREAM NILSTN :=
('*', '*', '*', NILUNE, NILUNE, NILVAR, NILSTM);

STREAM NILINS :=
('*', '*', NILUNI, NILUNI, NILVAR, NILSTM);

STREAM NILOUT :=
('&', '&', NILUNI, NILUNI, NILVAR, NILSTM);

VARIAB NILVAR :=
(SP, SP, SP, SP, 0, NILSTM, NILVAR, NILDAT);

VARIAB NILVAN :=
('*', '*', '*', '*', '*', 0, NILSTM, NILVAR, NILDAT);

DATAV NILDAT :=
(0, 0, 0, 0, '*', '*');

% INITIATION OF VARIABLES %

REF UNITDE FREEUN := UNI(1);
REF STREAM FREEST := STR(1);
REF VARIAB FREEVA := VAB(1);
INT COUNTV := 1;
INT COUNTI := 1;
BYTE C := '+';
ENDDATA;

%***************************************************************************%

ENT DATA TREES;

% INITIATION OF VARIABLES %

INT RTD := 3;
INT BT := 0;
INT FAST := 0;
INT FLAG := 0;
REF EVENT EAST := NILEV;
REF EVENT EST := NILEV;
REF GATE GST := NILGA;
REF BOUNDECO BCST := NILBC;
REF UNITDE UST := NILUNI;
REF STREAM SST := NILSTM;
REF VARIAB VST := NILVAR;
ENDDATA;

Fig. III.2.3.1
Fig. III.2.3.2 Memory layout used in this work
Fig. III.2.3.3 Part of the list structure for the topology of the two pipe and valve system example used in Chapter 4
BMT, TI: /SH=BAI, BMT
IOR, AR2, T/LB, SLB/LB: SLRFR: SLRFWF, ELB/LB
/
INITS=3
ACTFIL=2
ASG=TI: 1: 2: 3
PAR=GEN: 40000: 40000
FRI=80
TASK=*..BMT
COMMON=DATBAS: RW
LIBR=FTREES: RO

Fig. III.3.1 Task BMT indirect command file
Fig. III.3.2 Task BTR indirect command file
Fig. III.3.3 Task BTR overlay description
CAL,TI:/SH=BA1,CAL
T/LB,ELB/LB
/
UNITS=3
ACTFIL=2
ASG=TI:1:2:3
PAR=GEN:40202:40203
PRI=80
TASK=...CAL
COMMON=DATBASE:RW
LIB=FTREES:RO
//
DEB / SH = DEB/MP
UNIT S = 3
ACTFIL = 2
ASG = T1:1:2:3
PAR = GEN:40000:40020
PRI = 80
TASK = DEB
COMMON = DATBAS:RW
LIBR = FTrees:RO

Fig. III.3.5 Task DEB indirect command file
Fig. III.3.6 Task DEB overlay description
DES, TI:/SH=BAI, DES
IOR=AR2, T/LB, ELB/LB
/
UNITS=3
ACTFIL=2
ASG=TI:1:2:3
PAR=GENi:42000:40200
PRI=80
TASK=DES
COMMON=DATBAS:RW
LIBRA=F TRES:RO
//

Fig. III.3.7 Task DES indirect command file
LIT: TI: /SH=BA1, LIT
IOR, AR2, T/LB, E/LB/LB
/
UNITS=3
ACTFIL=2
ASG=TI: 1: 2: 3
FAR=GEN: 4000: 40000
PRI=80
TASK=...LIT
COMMON=DATABASE: RW
LIBR=FTREES: RO
//

Fig. III.3.8 Task LIT indirect command file
Fig. III.3.9 Task PMT indirect command file
Fig. III.3.10 Task PRI indirect command file
PTR, TI: /SH=BA1, PTR
I0R, AR2, TI/LB, ELB/LB
/ 
UNITS=3
ACTFIL=2
ASG=TI: 1: 2: 3
PAR=GEN: 40000: 40000
PRI=80
TASK=...PTR
COMMON=CATBAS: RW
LIBR=FTREES: RO
//

Fig. III.3.11 Task PTR indirect command file
REU, TI: /SH=BA1, REU
IOR, AR2, T/LB, EL B/LB
/
UNITS=3
ACTFIL=2
ASG=TI: 1:2:3
PAR=GEN: 42022:40020
PRI=50
TASK=...REU
COMMON=DATBAS: RW
LIBR=FTREES: RO
//
RFA TI: /SH=BA1 > RFA
10R AR2 > T/LB ELB/LB
/
UNITS=3
ACTFIL=2
ASG=TI: 1: 2: 3
PAR=GEN: 40000: 42000
PRI=80
TASK=RFA
COMMON=DATBASE: RW
LIBR=FTREES: RO
/

Fig. III3.13 Task RFA indirect command file
Fig. III 3.14 Task RTU indirect command file
RVA, TI: /SH=BA1, RVA
IOR, AR2, T/LB, LB/LB
/
UNIT$=3
ACTFIL=2
ASG=TI: 1: 2: 3
PAR=GEN: 40000: 40000
PRI=80
TASK=...RVA
COMMON=DATBAS: RW
LIBR=FTREES: RO
/

Fig. III.3.15  Task RVA indirect command file
LISTING

III.1 Input Data for Pipe's Minitrees (Task BMT)
TYPE OF UNIT(*•**% TO TERMINATE INPUT)?
PIPE%

TYPE OF EVENT(* TO TERMINATE INPUT)?
M VARIABLE ?
Q 3-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
FAULT(***% TO TERM. INPUT)?
HI%
TYPE OF GATE(2-OR, 1-AND, 2-EXOR)?
NOT ALLOWED CONDITIONS:
FAULT(***% TO TERM. INPUT)?
BLOCKAGE%
FAULT(***% TO TERM. INPUT)?
CLOSED%
FAULT(***% TO TERM. INPUT)?
SHUT%
FAULT(***% TO TERM. INPUT)?

TYPE OF EVENT(* TO TERMINATE INPUT)?
T VARIABLE ?
P 3-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
FAULT(***% TO TERM. INPUT)?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT)?
T VARIABLE ?
P 3-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
FAULT(***% TO TERM. INPUT)?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT)?
B VARIABLE ?
3-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
FAULT(***% TO TERM. INPUT)?
FL-EX-ENV?
PROBABILITY ?
0.0001

TYPE OF EVENT(* TO TERMINATE INPUT)?
M
VARiABLE ?
Q
3-OUT, 1-IN, 2-NO VAR., 3-I-VAL ?
0
FAULT(*#% TO TERM. INPUT) ?
0%
TYPE OF GATE(0=OR, 1-AND, 2-EXOR) ?
0
NOT ALLOWED CONDITIONS :
FAULT(*#% TO TERM. INPUT) ?
#**

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARiABLE ?
P
3-OUT, 1-IN, 2-NO VAR., 3-I-VAL ?
I
FAULT(*#% TO TERM. INPUT) ?
0%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARiABLE ?
P
3-OUT, 1-IN, 2-NO VAR., 3-I-VAL ?
1
FAULT(*#% TO TERM. INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B
VARiABLE ?
3-OUT, 1-IN, 2-NO VAR., 3-I-VAL ?
2
FAULT(*#% TO TERM. INPUT) ?
BLOCKAGE%
PROBABILITY ?
0.8282

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B
VARiABLE ?
3-OUT, 1-IN, 2-NO VAR., 3-I-VAL ?
2
FAULT(*#% TO TERM. INPUT) ?
LK-LP-ENV%
PROBABILITY ?
0.8225

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARiABLE ?

PAGE  584
PIPEID.FRO
NOT ALLOWED CONDITIONS:

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
Q
3-OUT,1-IN,2-NO VAR,3-1-VAR ?
1
FAULT(*%% TO TERM. INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
Q
3-OUT,1-IN,2-NO VAR,3-1-VAR ?
1
FAULT(*%% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B VARIABLE ?
Q
3-OUT,1-IN,2-NO VAR,3-1-VAR ?
2
FAULT(*%% TO TERM. INPUT) ?
BLOCKAGE
PROBABILITY ?
B.0003

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B VARIABLE ?
Q
3-OUT,1-IN,2-NO VAR,3-1-VAR ?
2
FAULT(*%% TO TERM. INPUT) ?
LK-HP-ENV?
PROBABILITY ?
B.0004

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
P
3-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(*00% TO TERM. INPUT) ?
LO%
TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?
3
NOT ALLOWED CONDITIONS :
FAULT(*00% TO TERM. INPUT) ?
RL-EX-ENV%
FAULT(*00% TO TERM. INPUT) ?
**%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
Q
3-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(*00% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B
VARIABLE ?
Q
3-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
2
FAULT(*00% TO TERM. INPUT) ?
LK-LP-ENV%
PROBABILITY ?
3.8885

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
T
3-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
3
FAULT(*00% TO TERM. INPUT) ?
HI%
TYPE OF GATE(3-OR, 1-AND, 2-EXOR) ?
3
NOT ALLOWED CONDITIONS :
FAULT(*00% TO TERM. INPUT) ?
TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
T
Ø-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
I
FAULT(*%% TO TERM. INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B
VARIABLE ?
Ø-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
2
FAULT(*%% TO TERM. INPUT) ?
EXT-FIRE%
PROBABILITY ?
3.0001

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
T
Ø-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
Ø
FAULT(*%% TO TERM. INPUT) ?
LO%
TYPE OF GATE(Ø-OR, 1-AND, 2-EXOR) ?
Ø
NOT ALLOWED CONDITIONS :
FAULT(*%% TO TERM. INPUT) ?
*%%
NOT ALLOWED CONDITIONS:

FAULT(**% TO TERM. INPUT) ?
**%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
X 0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(**% TO TERM. INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?
X 0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
O
FAULT(**% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
N VARIABLE ?
Q 0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
O
FAULT(**% TO TERM. INPUT) ?
NO-FLOW%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
GATE(0-OR, 1-AND, 2-EXOR)?
O
NOT ALLOWED CONDITIONS:

FAULT(**% TO TERM. INPUT) ?
**%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
Q
D-OUT, I-IN, 2-NO VAR., 3-I-VAR?
1
FAULT(* &% TO TERMINATE INPUT)?
NO-FLOW

TYPE OF EVENT(* TO TERMINATE INPUT)?
B
VARIABLE?
D-OUT, I-IN, 2-NO VAR., 3-I-VAR?
2
FAULT(* &% TO TERMINATE INPUT)?
COMP-BLOC?
PROBABILITY?
8.0001

TYPE OF EVENT(* TO TERMINATE INPUT)?
M
VARIABLE?
Q
D-OUT, I-IN, 2-NO VAR., 3-I-VAR?
S
FAULT(* &% TO TERMINATE INPUT)?
GT0
TYPE OF GATE(*-OR, I-AND, 2-EXOR)?
S
NOT ALLOWED CONDITIONS:
FAULT(* &% TO TERMINATE INPUT)?
***%

TYPE OF EVENT(* TO TERMINATE INPUT)?
T
VARIABLE?
Q
D-OUT, I-IN, 2-NO VAR., 3-I-VAR?
I
FAULT(* &% TO TERMINATE INPUT)?
GT0

TYPE OF EVENT(* TO TERMINATE INPUT)?
*
TYPE OF UNIT(* & & & & & TO TERMINATE INPUT)?
*****%
LISTING

III.2 Input Data for Heat-Exchanger Minitrees (Task BMT)
TYPE OF UNIT(*++++% TO TERMINATE INPUT) ?
HEAT-EX%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
T
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
G
FAULT(*+++% TO TERMINATE INPUT) ?
HI%

TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?
G
NOT ALLOWED CONDITIONS :

FAULT(*+++% TO TERMINATE INPUT) ?
*+++%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
T
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(*+++% TO TERMINATE INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
Q
VARIABLE ?
Q
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(*+++% TO TERMINATE INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
R
VARIABLE ?
R
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
2
FAULT(*+++% TO TERMINATE INPUT) ?
2-LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B
VARIABLE ?
B
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
G
FAULT(*+++% TO TERMINATE INPUT) ?
EXT-FIRE%
PROBABILITY ?
8.8881
TYPE OF EVENT(* TO TERMINATE INPUT) ? M
VARIABLE ?
T
0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
0
FAULT(*% TO TERM. INPUT) ? LO%
TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?
0
NOT ALLOWED CONDITIONS:
FAULT(*% TO TERM. INPUT) ? *%*

TYPE OF EVENT(* TO TERMINATE INPUT) ? T
VARIABLE ?
T
0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
1
FAULT(*% TO TERM. INPUT) ? LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ? R
VARIABLE ?
R
0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
2
FAULT(*% TO TERM. INPUT) ? HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ? R
VARIABLE ?
R
0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
2
FAULT(*% TO TERM. INPUT) ? HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ? M
VARIABLE ?
M
0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ?
2
FAULT(*% TO TERM. INPUT) ? HI%
TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?
2
NOT ALLOWED CONDITIONS:
FAULT(*% TO TERM. INPUT) ?
TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
Q 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ? 0'
FAULT(*8% TO TERM. INPUT) ? LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?
C 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ? 0'
FAULT(*8% TO TERM. INPUT) ? HI%
TYPE OF GATE(8-OR, 1-AND, 2-EXOR) ?
3 NOT ALLOWED CONDITIONS :
FAULT(*8% TO TERM. INPUT) ? *8%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
C 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ? 0'
FAULT(*8% TO TERM. INPUT) ? HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
R VARIABLE ?
C 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR ? 0'
FAULT(*8% TO TERM. INPUT) ? BI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
R

PAGE HEATID.PRO
VARIABLE

0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?

2

FAULT(*#% TO TERM. INPUT)

2=HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

B

VARIABLE

0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?

2

FAULT(*#% TO TERM. INPUT)

EXT=FLR%  
PROBABILITY ?

Z.0002

TYPE OF EVENT(* TO TERMINATE INPUT) ?

M

VARIABLE

0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?

2

FAULT(*#% TO TERM. INPUT)

TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?

2

NOT ALLOWED CONDITIONS:

FAULT(*#% TO TERM. INPUT)

TYPE OF EVENT(* TO TERMINATE INPUT) ?

T

VARIABLE

B

0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?

2

FAULT(*#% TO TERM. INPUT)

LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

R

VARIABLE

0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?

2

FAULT(*#% TO TERM. INPUT)

CZ

TYPE OF EVENT(* TO TERMINATE INPUT) ?

M

VARIABLE

0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
FAULT(*%*% TO TERM INPUT) ?
C

TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?
I

NOT ALLOWED CONDITIONS:

FAULT(*%*% TO TERM INPUT) ?
*%*%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
B
3-OUT, 1-IN, 2-NO VAR, 3=1-VAR ?
I

FAULT(*%*% TO TERM INPUT) ?
NO-FLOW%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
Q
3-OUT, 1-IN, 2-NO VAR, 3=1-VAR ?
I

FAULT(*%*% TO TERM INPUT) ?
G1%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
Q
3-OUT, 1-IN, 2-NO VAR, 3=1-VAR ?
I

FAULT(*%*% TO TERM INPUT) ?
G1%

TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?
I

NOT ALLOWED CONDITIONS:

FAULT(*%*% TO TERM INPUT) ?
*%*%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
Q
3-OUT, 1-IN, 2-NO VAR, 3=1-VAR ?
I

FAULT(*%*% TO TERM INPUT) ?
G1%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
C

PAGE

HEATID.PRO

----------
NOT ALLOWED CONDITIONS:

FAULT(*% TO TERM. INPUT) ?

TYPE OF EVENT(* TO TERMINATE INPUT) ?

FAULT(*% TO TERM. INPUT) ?

TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?

0
T
VARIABLE ?
T
O-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
S
FAULT(*%* TO TERMINATE INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
C
O-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
S
FAULT(*%* TO TERMINATE INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
O-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
S
FAULT(*%* TO TERMINATE INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
T
O-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
S
FAULT(*%* TO TERMINATE INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
C
O-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
S
FAULT(*%* TO TERMINATE INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?
P
598

0-OUT, 1-IN, 2-NO VAR, 3-1-VAR ?
1
FAULT(*% TO TERMINATE INPUT) ?
H% TYPE OF GATE(2-OR, 1-AND, 2-EXOR) ?
B
NOT ALLOWED CONDITIONS :

FAULT(*% TO TERMINATE INPUT) ?
LK-LEV-ENV% FAULT(*% TO TERMINATE INPUT) ?
**%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
0 0-OUT, 1-IN, 2-NO VAR, 3-1-VAR ?
1
FAULT(*% TO TERMINATE INPUT) ?
H% TYPE OF EVENT(* TO TERMINATE INPUT) ?
Q VARIABLE ?
0 0-OUT, 1-IN, 2-NO VAR, 3-1-VAR ?
1
FAULT(*% TO TERMINATE INPUT) ?
H% TYPE OF EVENT(* TO TERMINATE INPUT) ?
R VARIABLE ?
0 0-OUT, 1-IN, 2-NO VAR, 3-1-VAR ?
1
FAULT(*% TO TERMINATE INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B VARIABLE ?
0 0-OUT, 1-IN, 2-NO VAR, 3-1-VAR ?
1
FAULT(*% TO TERMINATE INPUT) ?
BLOCKAGE% PROBABILITY ?
3.3333

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?
0 0-OUT, 1-IN, 2-NO VAR, 3-1-VAR ?
1
FAULT(*% TO TERMINATE INPUT) ?
LK-LEV-ENV% PROBABILITY ?
3.3333

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?

PAGE HEATID: PRO
NOT ALLOWED CONDITIONS:

FAULT(*9*% TO TERM. INPUT)?
LO%

TYPE OF GATE(2-OR, 1-AND, 2-EXOR)?

FAULT(*9*% TO TERM. INPUT)?
FL-EX-ENV%
FAULT(*9*% TO TERM. INPUT)?
**%

TYPE OF EVENT(* TO TERMINATE INPUT)?
T VARIABLE?
Q

FAULT(*9*% TO TERM. INPUT)?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT)?
T VARIABLE?
Q

FAULT(*9*% TO TERM. INPUT)?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT)?
B VARIABLE?
Q

FAULT(*9*% TO TERM. INPUT)?
LK-IP-ENV%
PROBABILITY?
0.8885

TYPE OF EVENT(* TO TERMINATE INPUT)?
M VARIABLE?
A

FAULT(*9*% TO TERM. INPUT)?
HI%
TYPE OF GATE(2-OR, 1-AND, 2-EXOR)?

FAULT(*9*% TO TERM. INPUT)?
LK-IP-ENV%

PAGE HEATID.PRO

--------

-------
FAULT(*%# TO TERM. INPUT)?

TYPE OF EVENT(* TO TERMINATE INPUT)?

T VARIABLE?
B
O-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
I
FAULT(*%# TO TERM. INPUT)?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT)?

T VARIABLE?
B
O-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
I
FAULT(*%# TO TERM. INPUT)?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT)?

B VARIABLE?
O-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
I
FAULT(*%# TO TERM. INPUT)?
BLOCKAGE%
PROBABILITY?
.08886

TYPE OF EVENT(* TO TERMINATE INPUT)?

B VARIABLE?
O-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
I
FAULT(*%# TO TERM. INPUT)?
LK-HP-ENV%
PROBABILITY?
.08117

TYPE OF EVENT(* TO TERMINATE INPUT)?

M VARIABLE?
A
O-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
I
FAULT(*%# TO TERM. INPUT)?
LO%
TYPE OF GATE(0-OR, 1-AND, 2-EXOR)?

NOT ALLOWED CONDITIONS:
FAULT(*%# TO TERM. INPUT)?
TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
B 2-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
TFAULT(*0% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
B 2-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
TFAULT(*0% TO TERM. INPUT) ?
HI%  

TYPE OF EVENT(* TO TERMINATE INPUT) ?
B VARIABLE ?
2-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
2' TFAULT(*0% TO TERM. INPUT) ?
LK-LP-ENV%  
PROBABILITY ?
0.8887  

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?
Q 2-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
Q' TFAULT(*0% TO TERM. INPUT) ?
HI%  
TYPE OF GATE(0-OR, 1-AND, 2-EXOR) ?
0' TNOT ALLOWED CONDITIONS:  
TFAULT(*0% TO TERM. INPUT) ?
LOCKAGE%  
TFAULT(*0% TO TERM. INPUT) ?
CLOSED%  
TFAULT(*0% TO TERM. INPUT) ?
SHUT%  
TFAULT(*0% TO TERM. INPUT) ?
*0%  

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
1
FAULT(*\%\% TO TERM. INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
2
FAULT(*\%\% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
3
FAULT(*\%\% TO TERM. INPUT) ?
RL-EX-ENV

PROBABILITY ?
2 8888

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
4
FAULT(*\%\% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
5
FAULT(*\%\% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
6
FAULT(*\%\% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
7
FAULT(*\%\% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
8
FAULT(*\%\% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
9
FAULT(*\%\% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?

VARIABLE ?
P
\$\text{0-OUT, 1-IN, 2-NO }\text{VAR}, 3-1-\text{VAR} ?
A
FAULT(*\%\% TO TERM. INPUT) ?
LO%
TYPE OF EVENT(* TO TERMINATE INPUT) ?
B
VARIABLE ?

FAULT(*% TO TERM. INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M
VARIABLE ?

FAULT(*% TO TERM. INPUT) ?
HI%

NOT ALLOWED CONDITIONS :

FAULT(*% TO TERM. INPUT) ?
BLOCKAGE%
FAULT(*% TO TERM. INPUT) ?
CLOSED%
FAULT(*% TO TERM. INPUT) ?
SHUT%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T
VARIABLE ?

FAULT(*% TO TERM. INPUT) ?
HI%
TYPE OF EVENT(* TO TERMINATE INPUT)?
T VARIABLE?
A 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
9
FAULT(*% TO TERMIN. INPUT)?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT)?
B VARIABLE?
A 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
9
FAULT(*% TO TERMIN. INPUT)?
FL-EX-ENV% PROBABILITY,
3.0011

TYPE OF EVENT(* TO TERMINATE INPUT)?
M VARIABLE?
B 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
9
FAULT(*% TO TERMIN. INPUT)?
LO%
TYPE OF GATE(0-OR, 1-AND, 2-EXOR)?
9
NOT ALLOWED CONDITIONS:
FAULT(*% TO TERMIN. INPUT)?
*%*

TYPE OF EVENT(* TO TERMINATE INPUT)?
T VARIABLE?
A 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
I
FAULT(*% TO TERMIN. INPUT)?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT)?
T VARIABLE?
A 0-OUT, 1-IN, 2-NO VAR., 3-I-VAR?
9
FAULT(*% TO TERMIN. INPUT)?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT)?
B VARIABLE?

Type of event(* to terminate input) ?

B VARIABLE ?

@-OUT, I-IN, 2-NO VAR., 3-I-VAR ?

FAULT(*@% to term. input) ?

blockage%
probability ?

@.012

Type of event(* to terminate input) ?

M VARIABLE ?

@-OUT, I-IN, 2-NO VAR., 3-I-VAR ?

FAULT(*@% to term. input) ?

HI%

Type of gate(3-or, 1-and, 2-exor) ?

0

Not allowed conditions:

FAULT(*@% to term. input) ?

***%

Type of event(* to terminate input) ?

T VARIABLE ?

@-OUT, I-IN, 2-NO VAR., 3-I-VAR ?

I

FAULT(*@% to term. input) ?

Hi%

Type of event(* to terminate input) ?

M VARIABLE ?

@-OUT, I-IN, 2-NO VAR., 3-I-VAR ?

8

FAULT(*@% to term. input) ?

LO%

Type of gate(3-or, 1-and, 2-exor) ?

8

Not allowed conditions:

FAULT(*@% to term. input) ?

***%
TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
X
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(*% TO TERM. INPUT) ?
LO%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?
D
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(*% TO TERM. INPUT) ?
HI%

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?
D
0-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
I
FAULT(*% TO TERM. INPUT) ?
LO%
TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?
B 3-OUT, 1-IN, 2-NO VAR, 3-I-VAR ?
'...
FAULT(*% TO TERM. INPUT) ?
NO-FLOW%
TYPE OF GATE(3-OR, 1-AND, 2-EXOR) ?
0
NOT ALLOWED CONDITIONS :
TYPE OF EVENT(* TO TERMINATE INPUT) ?
B VARIABLE ?

TYPE OF EVENT(* TO TERMINATE INPUT) ?
M VARIABLE ?

TYPE OF EVENT(* TO TERMINATE INPUT) ?
T VARIABLE ?

TYPE OF EVENT(* TO TERMINATE INPUT) ?
* TYPE OF UNIT(*+++++% TO TERMINATE INPUT) ?
+++++++
LISTING

III.3 Input Data for a Two Pipe and Valve System (Task REU)
TOPOLOGY OF THE SYSTEM  PART I
****************************************

DESCRIPTION OF THE UNITS
---------------------------

HOW MANY UNITS IN THE SYSTEM ? 5

NO OF THE UNIT ?
1
TYPE OF UNIT(*****% TO TERMINATE INPUT) ? DUMMY-H%

NO OF THE UNIT ?
2
TYPE OF UNIT(*****% TO TERMINATE INPUT) ? PIPES%

NO OF THE UNIT ?
3
TYPE OF UNIT(*****% TO TERMINATE INPUT) ? VALVES%

NO OF THE UNIT ?
4
TYPE OF UNIT(*****% TO TERMINATE INPUT) ? PIPES%

NO OF THE UNIT ?
5
TYPE OF UNIT(*****% TO TERMINATE INPUT) ? DUMMY-T%
III.4 Input Data for a Two Pipe and Valve System (Task DES)
TOPOLOGY OF THE SYSTEM PART 2
**************************************

STREAMS AND VARIABLES OF EACH UNIT
-------------------------------------

UNIT NO. : 1  DUMMY=H
**************************

INPUT STREAMS
--------------

I/O STREAM(1-IN, 2-OUT, 2-DUMMY, ++TERMIN) ?
2
I/O STREAM(1-IN, 2-OUT, 2-DUMMY, ++TERMIN) ?

OUTPUT STREAMS
--------------

I/O STREAM(1-IN, 2-OUT, 2-DUMMY, ++TERMIN) ?
0
TO UNIT : ?
2

VARIABLE DESCRIPTION(Y/N) ?
Y
NAME OF VARIABLE ?
P
NO. OF THE VARIABLE ?
1
IS THE VARIABLE MEASURED(Y/N) ?
N
VARIABLE DESCRIPTION(Y/N) ?
Y
NAME OF VARIABLE ?
Q
NO. OF THE VARIABLE ?
1
IS THE VARIABLE MEASURED(Y/N) ?
N
VARIABLE DESCRIPTION(Y/N) ?
N
I/O STREAM(1-IN, 2-OUT, 2-DUMMY, ++TERMIN) ?

UNIT NO. : 2  PIPE
****************************

INPUT STREAMS
-------------
I/O STREAM(1-IN,0-OUT,2-DUMMY,+TERMIN) ?
1
FROM UNIT : ?
1

VARIABLE DESCRIPTION(Y/N) ?
Y
IS THIS AN INTERNAL VAR.(Y/N) ?
N
NAME OF VARIABLE ?
P
NAME OF VARIABLE ?
Q
VARIABLE DESCRIPTION(Y/N) ?
N
I/O STREAM(1-IN,0-OUT,2-DUMMY,+TERMIN) ?
+

OUTPUT STREAMS
--------------

I/O STREAM(1-IN,0-OUT,2-DUMMY,+TERMIN) ?
0
TO UNIT : ?
3

VARIABLE DESCRIPTION(Y/N) ?
Y
NAME OF VARIABLE ?
P
NO. OF THE VARIABLE ?
2
IS THE VARIABLE MEASURED(Y/N) ?
N
VARIABLE DESCRIPTION(Y/N) ?
Y
NAME OF VARIABLE ?
Q
NO. OF THE VARIABLE ?
2
IS THE VARIABLE MEASURED(Y/N) ?
N
VARIABLE DESCRIPTION(Y/N) ?
N
I/O STREAM(1-IN,0-OUT,2-DUMMY,+TERMIN) ?
+

UNIT NO. : 3 VALVE
*****************************

INPUT STREAMS
-------------

PAGE DES .PRO
VAR DESCRIPTION (Y/N) ?
Y
IS THIS AN INTERNAL VAR. (Y/N) ?
N
NAME OF VARIABLE ?
P
NAME OF VARIABLE ?
Q
VAR DESCRIPTION (Y/N) ?
N
I/O STREAM (1-IN, 0-OUT, 2-DUMMY, +-TERMIN) ?
+

OUTPUT STREAMS

----------------------------------------
I/O STREAM (1-IN, 0-OUT, 2-DUMMY, +-TERMIN) ?
+
TO UNIT : ?
4

VAR DESCRIPTION (Y/N) ?
Y
NAME OF VARIABLE ?
P
NO. OF THE VARIABLE ?
3
IS THE VARIABLE MEASURED (Y/N) ?
Y
HIGH ?
100
LOW ?
0
PRIORITY OF THE VAR. ?
1
VAR DESCRIPTION (Y/N) ?
Y
NAME OF VARIABLE ?
Q
NO. OF THE VARIABLE ?
3
IS THE VARIABLE MEASURED (Y/N) ?
N
VAR DESCRIPTION (Y/N) ?
N
I/O STREAM (1-IN, 0-OUT, 2-DUMMY, +-TERMIN) ?
+

UNIT NO. : 4 PIPE

INPUT STREAMS

PAGE DES PRO

-------------
I/O STREAM(1-IN,2-OUT,2-DUMMY,+TERMIN) ?
FROM UNIT : ?
3

VARIABLE DESCRIPTION(Y/N) ?
Y
IS THIS AN INTERNAL VAR.(Y/N) ?
N
NAME OF VARIABLE ?
P
NAME OF VARIABLE ?
Q
VARIABLE DESCRIPTION(Y/N) ?
N
I/O STREAM(1-IN,2-OUT,2-DUMMY,+TERMIN) ?
+

OUTPUT STREAMS
------------------

I/O STREAM(1-IN,2-OUT,2-DUMMY,+TERMIN) ?
3
TO UNIT : ?
5

VARIABLE DESCRIPTION(Y/N) ?
Y
NAME OF VARIABLE ?
P
NAME OF VARIABLE ?
Q
NO. OF THE VARIABLE ?
4
IS THE VARIABLE MEASURED(Y/N) ?
N
VARIABLE DESCRIPTION(Y/N) ?
Y
NAME OF VARIABLE ?
P
NAME OF VARIABLE ?
Q
NO. OF THE VARIABLE ?
4
IS THE VARIABLE MEASURED(Y/N) ?
Y
HIGH ?
180
LOW ?
0
PRIORITY OF THE VAR. ?
2
VARIABLE DESCRIPTION(Y/N) ?
N
I/O STREAM(1-IN,2-OUT,2-DUMMY,+TERMIN) ?
+

UNIT NO. : 5  DUMMY-T

PAGE  DES  *PRO
-------
INPUT STREAMS
-------------

I/O STREAM (1-IN, 0-OUT, 2-DUMMY, +TERMIN) ?
1
FROM UNIT : ?
4

VARIABLE DESCRIPTION (Y/N) ?
Y
IS THIS AN INTERNAL VAR. (Y/N) ?
N
NAME OF VARIABLE ?
P
NAME OF VARIABLE ?
Q
VARIABLE DESCRIPTION (Y/N) ?
N
I/O STREAM (1-IN, 2-OUT, 2-DUMMY, +TERMIN) ?
+

OUTPUT STREAMS
---------------

I/O STREAM (1-IN, 2-OUT, 2-DUMMY, +TERMIN) ?
2
I/O STREAM (1-IN, 2-OUT, 2-DUMMY, +TERMIN) ?
+
III.5 Output of a Two Pipe and Valve System (Task DEB)
ARRAY OF BOUND. COND.

**************************

B. C. : 1
VARBC : 1
NA1 :
  - LO
    CLOSED
    BLOCKAGE
    SHUT
NA2 :
NA3 :
NEXTBC: 2

B. C. : 2
VARBC : 3
NA1 :
  - HI
NA2 :
NA3 :
NEXTBC: 3

B. C. : 3
VARBC : 5
NA1 :
  - LO
NA2 :
  OTHER-CAU
NA3 :
NEXTBC: 4

B. C. : 4
VARBC : 7
NA1 :
  - HI
NA2 :
  OTHER-CAU
NA3 :
NEXTBC: 5

B. C. : 5
VARBC : 9
NA1 :
  - LO
NA2 :
  OTHER-CAU
NA3 :
NEXTBC: 6

B. C. : 6
VARBC : 11
NA1 :

PAGE
NA2
   OTHER-CAU
NA3:
NEXTBC: 7

BA C*: 7
VARBC: 13
NA1:
   LO
NA2:
   OTHER-CAU
NA3:
NEXTBC: 8

BA C*: 8
VARBC: 15
NA1:
   HI
NA2:
   OTHER-CAU
NA3:
NEXTBC: 9

BA C*: 9
VARBC: 17
NA1:
   OTHER-CAU
NA2:
NA3:
NEXTBC: 10

BA C*: 10
VARBC: 19
NA1:
   OTHER-CAU
NA2:
NA3:
NEXTBC: 11

BA C*: 11
VARBC: 21
NA1:
   LO
   LK-LP-ENV
NA2:
NA3:
NEXTBC: 12

BA C*: 12
VARBC: 23
NA1:
   HI
   FL-EX-ENV
NA2:
BoC: 19
VABBC: 48
NA1: LO
NA2: NA3:
NEXTBC: 20

BoC: 20
VABBC: 50
NA1: HI
NA2: NA3:
NEXTBC: 21

BoC: 21
VABBC: 52
NA1: NA2: NA3:
NEXTBC: 22

BoC: 22
VABBC: 55
NA1: NA2: NA3:
NEXTBC: 23

BoC: 23
VABBC: 57
NA1: LO
CLOSED BLOCKAGE SHUT
NA2: NA3:
NEXTBC: 24

BoC: 24
VABBC: 62
NA1: HI OPEN WIDE-OPEN
NA2: FAIL-OPEN
NA3:
NEXTBC: 25

BoC: 25
<table>
<thead>
<tr>
<th>NEXTBC:</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC:</td>
<td>32</td>
</tr>
<tr>
<td>VARBC:</td>
<td>89</td>
</tr>
<tr>
<td>NA1:</td>
<td></td>
</tr>
<tr>
<td>NA2:</td>
<td></td>
</tr>
<tr>
<td>NA3:</td>
<td></td>
</tr>
<tr>
<td>NEXTBC:</td>
<td>33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NEXTBC:</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC:</td>
<td>33</td>
</tr>
<tr>
<td>VARBC:</td>
<td>92</td>
</tr>
<tr>
<td>NA1:</td>
<td></td>
</tr>
<tr>
<td>NA2:</td>
<td></td>
</tr>
<tr>
<td>NA3:</td>
<td></td>
</tr>
<tr>
<td>NEXTBC:</td>
<td>34</td>
</tr>
</tbody>
</table>
ARRAY OF BOUND. COND.
***********************

B. C. : 34
VARBC : 94
NA1 : 
   HI
NA2 :
NA3 :
NEXTBC: 35

B. C. : 35
VARBC : 95
NA1 :
   HI
   FL-EX-ENV
NA2 :
NA3 :
NEXTBC: 36

B. C. : 36
VARBC : 96
NA1 :
   HI
   OPEN
   FL-EX-ENV
   WIDE-OPEN
NA2 :
   FAIL-OPEN
NA3 :
NEXTBC: 37

B. C. : 37
VARBC : 97
NA1 :
   HI
   OPEN
   FL-EX-ENV
   WIDE-OPEN
NA2 :
   FAIL-OPEN
NA3 :
NEXTBC: 38

B. C. : 38
VARBC : 98
NA1 :
   HI
   OPEN
   FL-EX-ENV
   WIDE-OPEN
NA2 :
   FAIL-OPEN
NA3 :
NEXTBC: 39

B. Co. : 39
VARBC : 99
NA1 :
  HI
  OPEN
  FL-EX-ENV
  WIDE-OPEN

NA2 :
  FAIL-OPEN
NA3 :
NEXTBC: 40

B. Co. : 40
VARBC : 105
NA1 :
  HI
  OPEN
  FL-EX-ENV
  WIDE-OPEN

NA2 :
  FAIL-OPEN
NA3 :
NEXTBC: 41
ARRAY OF EVENTS

***************

EVENT : 1
IDEN: M
UNID: 6 DUMMY-H
VAR: Q
INOU: 0002003
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 2
TOP: 3
BACKG: NIL GATE
NEXTG: 1
UNNO: NIL UNIT
PROB: 0.0020

EVENT : 2
IDEN: T
UNID: 6 DUMMY-H
VAR: P
INOU: 0000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 3
TOP: NIL EVENT
BACKG: 1
NEXTG: GM
UNNO: NIL UNIT
PROB: 0.0020

EVENT : 3
IDEN: M
UNID: 6 DUMMY-H
VAR: Q
INOU: 0000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 4
TOP: 5
BACKG: NIL GATE
NEXTG: 2
UNNO: NIL UNIT
PROB: 0.0020

EVENT : 4
IDEN: T
UNID: 6 DUMMY-H
VAR: P
INOU: 0000000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 5
TOP: NIL EVENT
EVENT : 5
IDENT: M
UNID: 6 DUMMY=H
VAR: W
INOU: 02000000
NAPA: NIL VARIABLE
FAULT: HI
NEXT: 6
TOP: 7
BACKG: NIL GATE
NEXTG: 3
UNNO: NIL UNIT
PROB: 0.0000

EVENT : 6
IDENT: B
UNID: 6 DUMMY=H
VAR: S
INOU: 02000000
NAPA: NIL VARIABLE
FAULT: OTHER-CAU
NEXT: 7
TOP: NIL EVENT
BACKG: 3
NEXTG: NIL GATE
UNNO: NIL UNIT
PROB: 0.0000

EVENT : 7
IDENT: M
UNID: 6 DUMMY=H
VAR: W
INOU: 02000000
NAPA: NIL VARIABLE
FAULT: LO
NEXT: 8
TOP: 9
BACKG: NIL GATE
NEXTG: 4
UNNO: NIL UNIT
PROB: 0.0000

EVENT : 8
IDENT: B
UNID: 6 DUMMY=H
VAR: S
INOU: 02000000
NAPA: NIL VARIABLE
FAULT: OTHER-CAU
NEXT: 9
TOP: NIL EVENT
BACKG: 4
EVENT: 9
IDEN: M
UNID: 6 DUMMY-H
VAR: T
INOU: 02200000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 10
TOP: 11
BACKG: NIL GATE
NEXTG: 5
UNN: NIL UNIT
PROB: 0.0000

EVENT: 10
IDEN: B
UNID: 6 DUMMY-H
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: OTHER-CAU
NEXT: 11
TOP: NIL EVENT
BACKG: 5
NEXTG: NIL GATE
UNN: NIL UNIT
PROB: 0.0001

EVENT: 11
IDEN: M
UNID: 6 DUMMY-H
VAR: T
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 12
TOP: 13
BACKG: NIL GATE
NEXTG: 6
UNN: NIL UNIT
PROB: 0.0000

EVENT: 12
IDEN: B
UNID: 6 DUMMY-H
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: OTHER-CAU
NEXT: 13
TOP: NIL EVENT
BACKG: 6
NEXTG: NIL GATE
EVENT : 13
IDEN : M
UNID : 6 DUMMY-H
VAR : X
INOU : 00000000
NAVAT : NIL VARIABLE
FAULT : HI
NEXT : 14
TOP : 15
BACKG : NIL GATE
NEXTG : 7
UNNO : NIL UNIT
PROB : 0.0000

EVENT : 14
IDEN : B
UNID : 6 DUMMY-H
VAR : S
INOU : 02020000
NAVAT : NIL VARIABLE
FAULT : OTHER-CAU
NEXT : 15
TOP : NIL EVENT
BACKG : 7
NEXTG : NIL GATE
UNNO : NIL UNIT
PROB : 0.0000

EVENT : 15
IDEN : M
UNID : 6 DUMMY-H
VAR : X
INOU : 02020000
NAVAT : NIL VARIABLE
FAULT : LO
NEXT : 16
TOP : 17
BACKG : NIL GATE
NEXTG : 8
UNNO : NIL UNIT
PROB : 0.0000

EVENT : 16
IDEN : B
UNID : 6 DUMMY-H
VAR : S
INOU : 02020000
NAVAT : NIL VARIABLE
FAULT : OTHER-CAU
NEXT : 17
TOP : NIL EVENT
BACKG : 8
NEXTG : NIL GATE
UNNO : NIL UNIT
EVENT : 17
IDEN: M
UNID : 6  DUMMY-H
VAR: Q
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: NO-FLOW
NEXT: 18
TOP: 19
BACKG: NIL GATE
NEXTG: 9
UNN$: NIL UNIT
PROB : 0.3000

EVENT : 18
IDEN: B
UNID : 6  DUMMY-H
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: OTHER-CAU
NEXT: 19
TOP: NIL EVENT
BACKG: 9
NEXTG: NIL GATE
UNN$: NIL UNIT
PROB : 0.3000

EVENT : 19
IDEN: M
UNID : 6  DUMMY-H
VAR: Q
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: GT0
NEXT: 22
TOP: NIL EVENT
BACKG: NIL GATE
NEXTG: 18
UNN$: NIL UNIT
PROB : 3.0000

EVENT : 20
IDEN: B
UNID : 6  DUMMY-H
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: OTHER-CAU
NEXT: 21
TOP: NIL EVENT
BACKG: 18
NEXTG: NIL GATE
UNN$: NIL UNIT
PROB : 0.3000
EVENT : 21
IDEN: M
UNID: 7 DUMMY-T
VAR: P
INOU: 10202220
NAVAl: NIL VARIABLE
FAULT: HI
NEXT: 22
TOP: 23
BACKG: NIL GATE
NEXTG: 11
UNNG: NIL UNIT
PROB : 0.0000

EVENT : 22
IDEN: T
UNID: 7 DUMMY-T
VAR: Q
INOU: 10000000
NAVAl: NIL VARIABLE
FAULT: HI
NEXT: 23
TOP: NIL EVENT
BACKG: 11
NEXTG: QM
UNNG: NIL UNIT
PROB : 0.0000

EVENT : 23
IDEN: M
UNID: 7 DUMMY-T
VAR: P
INOU: 10202220
NAVAl: NIL VARIABLE
FAULT: LO
NEXT: 24
TOP: NIL EVENT
BACKG: NIL GATE
NEXTG: 12
UNNG: NIL UNIT
PROB : 0.0000

EVENT : 24
IDEN: T
UNID: 7 DUMMY-T
VAR: Q
INOU: 10000000
NAVAl: NIL VARIABLE
FAULT: LO
NEXT: 25
TOP: NIL EVENT
BACKG: 12
NEXTG: QM
UNNG: NIL UNIT
PROB : 0.0000
EVENT : 25
IDEN: M
UNID: 9 PIPE
VAR: Q
INQU: 36800020
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 26
TOP: 29
BACKG: NIL GATE
NEXTG: 13
UNNO: NIL UNIT
PROB : 0:0000

EVENT : 26
IDEN: T
UNID: 9 PIPE
VAR: P
INQU: 12800000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 27
TOP: NIL EVENT
BACKG: 13
NEXTG: QM
UNNO: NIL UNIT
PROB : 0:0000

EVENT : 27
IDEN: T
UNID: 9 PIPE
VAR: P
INQU: 00800000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 28
TOP: NIL EVENT
BACKG: 13
NEXTG: QM
UNNO: NIL UNIT
PROB : 0:0000

EVENT : 28
IDEN: B
UNID: 9 PIPE
VAR: S
INQU: 02000000
NAVA: NIL VARIABLE
FAULT: FL-EX-ENV
NEXT: 29 " "
TOP: NIL EVENT
BACKG: 13
NEXTG: NIL GATE
UNNO: NIL UNIT
PROB : 0:0001
EVENT : 29
IDEN: M
UNID: 9 PIPE
VAR: Q
INOU: 2000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 30
TOP: 34
BACKG: NIL GATE
NEXTG: 14
UNNS: NIL UNIT
PROB : 0.0023

EVENT : 30
IDEN: T
UNID: 9 PIPE
VAR: P
INOU: 1000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 31
TOP: NIL EVENT
BACKG: 14
NEXTG: QM
UNNS: NIL UNIT
PROB : 0.0000

EVENT : 31
IDEN: T
UNID: 9 PIPE
VAR: P
INOU: 3200000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 32
TOP: NIL EVENT
BACKG: 14
NEXTG: QM
UNNS: NIL UNIT
PROB : 0.0020

EVENT : 32
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 3200000
NAVA: NIL VARIABLE
FAULT: BLOCKAGE
NEXT: 33
TOP: NIL EVENT
BACKG: 14
NEXTG: NIL GATE
UNNS: NIL UNIT
PROB : 0.0002

EVENT : 33
EVENT: 34
IDEN: B
UNID: 9 PIPE
VAR: S
INQ: 02000000
NAVAR: NIL VARIABLE
FAULT: LO-LP-ENV
NEXT: 34
TOP: NIL EVENT
BACKG: 14
NEXTG: NIL GATE
UNNO: NIL UNIT
FROB: 0.0225

EVENT: 35
IDEN: M
UNID: 9 PIPE
VAR: P
INQ: 10000000
NAVAR: NIL VARIABLE
FAULT: HI
NEXT: 35
TOP: 39
BACKG: NIL GATE
NEXTG: 15
UNNO: NIL UNIT
FROB: 0.0000

EVENT: 36
IDEN: T
UNID: 9 PIPE
VAR: Q
INQ: 13000000
NAVAR: NIL VARIABLE
FAULT: HI
NEXT: 36
TOP: NIL EVENT
BACKG: 15
NEXTG: QM
UNNO: NIL UNIT
FROB: 0.0000

EVENT: 37
IDEN: B
EVENT : 38
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 02000003
NAVA: NIL VARIABLE
FAULT: H-HP-ENV
NEXT: 39
TOP: NIL EVENT
BACKG: 15
NEXTG: NIL GATE
UNNO: NIL UNIT
PROB: 0.0203

EVENT : 39
IDEN: M
UNID: 9 PIPE
VAR: P
INOU: 12000000
NAVA: NIL VARIABLE
FAULT: L0
NEXT: 43
TOP: 43
BACKG: NIL GATE
NEXTG: 16
UNNO: NIL UNIT
PROB: 0.0224

EVENT : 40
IDEN: T
UNID: 9 PIPE
VAR: Q
INOU: 20000000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 41
TOP: NIL EVENT
BACKG: 16
NEXTG: QM
UNNO: NIL UNIT
PROB: 2.0888

EVENT : 41
IDEN: T
UNID: 9 PIPE
VAR: Q  
INOU: 10000000  
NAV: NIL VARIABLE  
FAULT: LO  
NEXT: 42  
TOP: NIL EVENT  
BACKG: 16  
NEXTG: QM  
UNNO: NIL UNIT  
PROB: 0.0000

EVENT: 42  
IDEN: B  
UNID: 9 PIPE  
VAR: S  
INOU: 02000000  
NAV: NIL VARIABLE  
FAULT: LK-LP-ENV  
NEXT: 43  
TOP: NIL EVENT  
BACKG: 16  
NEXTG: NIL GATE  
UNNO: NIL UNIT  
PROB: 0.0005

EVENT: 43  
IDEN: M  
UNID: 9 PIPE  
VAR: T  
INOU: 02000000  
NAV: NIL VARIABLE  
FAULT: HI  
NEXT: 44  
TOP: 46  
BACKG: NIL GATE  
NEXTG: 17  
UNNO: NIL UNIT  
PROB: 0.0020

EVENT: 44  
IDEN: T  
UNID: 9 PIPE  
VAR: T  
INOU: 10000000  
NAV: NIL VARIABLE  
FAULT: HI  
NEXT: 45  
TOP: NIL EVENT  
BACKG: 17  
NEXTG: QM  
UNNO: NIL UNIT  
PROB: 0.0020

EVENT: 45  
IDEN: B  
UNID: 9 PIPE  
VAR: S
EVENT: 46
IDEN: M
UNID: 9 PIPE
VAR: T
INO: 02000000
NAVA: NIL VARIABLE
FAULT: L0
NEXT: 47
TOP: 48
BACKG: NIL GATE
NEXTG: 18
UNNO: NIL UNIT
PROB: 0.0000

EVENT: 47
IDEN: T
UNID: 9 PIPE
VAR: T
INO: 12000000
NAVA: NIL VARIABLE
FAULT: L0
NEXT: 48
TOP: 49
BACKG: NIL GATE
NEXTG: 19
UNNO: NIL UNIT
PROB: 0.0000

EVENT: 48
IDEN: M
UNID: 9 PIPE
VAR: X
INO: 02000000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 49
TOP: 50
BACKG: NIL GATE
NEXTG: 19
UNNO: NIL UNIT
PROB: 0.0000

EVENT: 49
IDEN: T
UNID: 9 PIPE
VAR: X
INO: 10000000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 50
TOP: NIL EVENT
BACKG: QM
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 50
IDEN: M
UNID : 9 PIPE
VAR: X
INQU: 00000003
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 51
TOP: 52
BACKG: NIL GATE
NEXTG: 20
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 51
IDEN: T
UNID : 9 PIPE
VAR: X
INQU: 00000003
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 52
TOP: NIL EVENT
BACKG: 20
NEXTG: QM
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 52
IDEN: M
UNID : 9 PIPE
VAR: Q
INQU: 00000020
NAVA: NIL VARIABLE
FAULT: NO-FLOW
NEXT: 53
TOP: 55
BACKG: NIL GATE
NEXTG: 21
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 53
IDEN: T
UNID : 9 PIPE
VAR: Q
INQU: 10000000
NAVA: NIL VARIABLE
FAULT: NO-FLOW
NEXT: 54
TOP: NIL EVENT
BACKG: 21
NEXTG: QM
UNN0: NIL UNIT
PROB: 0.0000

EVENT: 54
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 02000000
NAVAl: NIL VARIABLE
FAULT: COMP-BLOC
NEXT: 55
TOP: NIL EVENT
BACKG: 21
NEXTG: NIL GATE
UNN0: NIL UNIT
PROB: 0.0001

EVENT: 55
IDEN: M
UNID: 9 PIPE
VAR: Q
INOU: 00000000
NAVAl: NIL VARIABLE
FAULT: GT0
NEXT: 56
TOP: NIL EVENT
BACKG: NIL GATE
NEXTG: 22
UNN0: NIL UNIT
PROB: 0.0000

EVENT: 56
IDEN: T
UNID: 9 PIPE
VAR: Q
INOU: 10000000
NAVAl: NIL VARIABLE
FAULT: GT0
NEXT: 57
TOP: NIL EVENT
BACKG: 22
NEXTG: QM
UNN0: NIL UNIT
PROB: 0.0000

EVENT: 57
IDEN: M
UNID: 13 VALUE
VAR: Q
INOU: 00000000
NAVAl: NIL VARIABLE
FAULT: HI
EVENT : 58
IDEN: T
UNID: 13 VALVE
VAR: P
INO: 18008000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 59
TOP: NIL EVENT
BACKG: 23
NEXTG: 8M
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 59
IDEN: T
UNID: 13 VALVE
VAR: P
INO: 00000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 60
TOP: NIL EVENT
BACKG: 23
NEXTG: 8M
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 60
IDEN: B
UNID: 13 VALVE
VAR: S
INO: 02000080
NAVA: NIL VARIABLE
FAULT: WIDE-OPEN
NEXT: 61
TOP: NIL EVENT
BACKG: 23
NEXTG: NIL GATE
UNNO: NIL UNIT
PROB : 0.0001

EVENT : 61
IDEN: B
UNID: 13 VALVE
VAR: S
INO: 02000080
NAVA: NIL VARIABLE
FAULT: FL-EX-ENV
NEXT: 62
EVENT : 62
IDEN: M
UNID: 13 VALUE
VAR: Q
INOU: 00000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 63
TOP: 67
BACKG: NIL GATE
NEXTG: 24
UNNO: NIL UNIT
PROB: 0.0000

EVENT : 63
IDEN: T
UNID: 13 VALUE
VAR: P
INOU: 00000000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 64
TOP: NIL EVENT
BACKG: 24
NEXTG: QM
UNNO: NIL UNIT
PROB: 0.0000

EVENT : 64
IDEN: T
UNID: 13 VALUE
VAR: P
INOU: 10000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 65
TOP: NIL EVENT
BACKG: 24
NEXTG: QM
UNNO: NIL UNIT
PROB: 0.0000

EVENT : 65
IDEN: R
UNID: 13 VALUE
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: CLOSED
NEXT: 66
TOP: NIL EVENT
EVENT : 66
IDENT: B
UNID: 13 VALUE
VAR: S
INOU: 02330223
NAVA: NIL VARIABLE
FAULT: LK-LP-ENV
NEXT: 67
TOP: NIL EVENT
BACKG: 24
NEXTG: NIL GATE
UNNO: NIL UNIT
PROP: 0.0023

EVENT : 67
IDENT: M
UNID: 13 VALUE
VAR: S
INOU: 02330223
NAVA: NIL VARIABLE
FAULT: CLOSED
NEXT: 68
TOP: 70
BACKG: NIL GATE
NEXTG: 25
UNNO: NIL UNIT
PROP: 0.0033

EVENT : 68
IDENT: B
UNID: 13 VALUE
VAR: S
INOU: 22222222
NAVA: NIL VARIABLE
FAULT: BLOCKAGE
NEXT: 69
TOP: NIL EVENT
BACKG: 25
NEXTG: NIL GATE
UNNO: NIL UNIT
PROP: 0.0004

EVENT : 69
IDENT: B
UNID: 13 VALUE
VAR: S
INOU: 02330233
NAVA: NIL VARIABLE
FAULT: SHUT
NEXT: 70
TOP: NIL EVENT
BACKG: 25
NEXTG: NIL GATE
UNN0: NIL UNIT
PROB : 0.0005

EVENT : 70
IDENT : M
UNIT : 13 VALUE
VAR: P*
INOU: 10022000
NAVAl: NIL VARIABLE
FAULT: HI
NEXT: 71
TOP: 75'
BACKG: NIL GATE
NEXTG: 26
UNN0: NIL UNIT
PROB : 0.0000

EVENT : 71
IDENT : T
UNIT : 13 VALUE
VAR: Q*
INOU: 10022000
NAVAl: NIL VARIABLE
FAULT: HI
NEXT: 72
TOP: NIL EVENT
BACKG: 26
NEXTG: QM
UNN0: NIL UNIT
PROB : 0.0000

EVENT : 72
IDENT : T
UNIT : 13 VALUE
VAR: Q*
INOU: 00000000
NAVAl: NIL VARIABLE
FAULT: LO
NEXT: 73
TOP: NIL EVENT
BACKG: 26
NEXTG: QM
UNN0: NIL UNIT
PROB : 0.0000

EVENT : 73
IDENT: R
UNIT : 13 VALUE
VAR: S*
INOU: 02000000
NAVAl: NIL VARIABLE
FAULT: CLOSED
NEXT: 74
TOP: NIL EVENT
BACKG: 26
NEXTG: QM
EVENT: 74
IDEN: B
UNID: 13
VAR: S
INOU: 02000000
NAV: NIL VARIABLE
FAULT: LK-HP-ENV
NEXT: 75
TOP: NIL EVENT
BACKG: 26
NEXTG: NIL GATE
UNNO: NIL UNIT
PROB: 0.0002

EVENT: 75
IDEN: M
UNID: 13
VAR: P
INOU: 10000000
NAV: NIL VARIABLE
FAULT: LO
NEXT: 76
TOP: 60
BACKG: NIL GATE
NEXTG: 27
UNNO: NIL UNIT
PROB: 0.0002

EVENT: 76
IDEN: T
UNID: 13
VAR: Q
INOU: 02000000
NAV: NIL VARIABLE
FAULT: HI
NEXT: 77
TOP: NIL EVENT
BACKG: 27
NEXTG: QM
UNNO: NIL UNIT
PROB: 0.0002

EVENT: 77
IDEN: T
UNID: 13
VAR: Q
INOU: 10000000
NAV: NIL VARIABLE
FAULT: LO
NEXT: 78
TOP: NIL EVENT
BACKG: 27
NEXTG: QM
UNNO: NIL UNIT
<table>
<thead>
<tr>
<th>EVENT</th>
<th>IDEN</th>
<th>UNID</th>
<th>VALUE</th>
<th>VAR</th>
<th>INOU</th>
<th>NAVA</th>
<th>FAULT</th>
<th>NEXT</th>
<th>TOP</th>
<th>BACKG</th>
<th>NEXTG</th>
<th>UNNO</th>
<th>PROB</th>
</tr>
</thead>
<tbody>
<tr>
<td>78</td>
<td>B</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>NIL</td>
<td>LK-LP-ENV</td>
<td>79</td>
<td>NIL</td>
<td>27</td>
<td>NIL GATE</td>
<td>NIL UNIT</td>
<td>0.0000</td>
</tr>
<tr>
<td>79</td>
<td>B</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>NIL</td>
<td>WIDE-OPEN</td>
<td>80</td>
<td>NIL</td>
<td>27</td>
<td>NIL GATE</td>
<td>NIL UNIT</td>
<td>0.0000</td>
</tr>
<tr>
<td>80</td>
<td>M</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>NIL</td>
<td>HI</td>
<td>81</td>
<td>83</td>
<td>NIL GATE</td>
<td>28</td>
<td>NIL UNIT</td>
<td>0.0000</td>
</tr>
<tr>
<td>81</td>
<td>T</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>NIL</td>
<td>HI</td>
<td>82</td>
<td>83</td>
<td>NIL GATE</td>
<td>28</td>
<td>NIL UNIT</td>
<td>0.0000</td>
</tr>
</tbody>
</table>
EVENT: 82
IDEN: B
UNID: 13  VALVE
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: EXT-FIRE
NEXT: 83
TOP: NIL EVENT
BACKG: 28
NEXTG: NIL GATE
UNN0: NIL UNIT
PROB: 0.0001

EVENT: 83
IDEN: M
UNID: 13  VALVE
VAR: T
INOU: 00000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 84
TOP: 85
BACKG: NIL GATE
NEXTG: 29
UNN0: NIL UNIT
PROB: 0.0020

EVENT: 84
IDEN: T
UNID: 13  VALVE
VAR: T
INOU: 10000000
NAVA: NIL VARIABLE
FAULT: LO
NEXT: 85
TOP: NIL EVENT
BACKG: 29
NEXTG: QM
UNN0: NIL UNIT
PROB: 0.0000

EVENT: 85
IDEN: M
UNID: 13  VALVE
VAR: X
INOU: 00000000
NAVA: NIL VARIABLE
FAULT: HI
NEXT: 86
TOP: 87
BACKG: NIL GATE
NEXTG: 30
UNN0: NIL UNIT
PROB: 0.0020
EVENT 86
IDEN: T
UNID: 13 VALVE
VAR: X
INOI: 10000000
NAVAl: NIL VARIABLE
FAULT: HI
NEXT: 87
TOP: NIL EVENT
BACKG: 30
NEXTG: QM
UNNO: NIL UNIT
PROB: 0.0000

EVENT 87
IDEN: M
UNID: 13 VALVE
VAR: X
INOI: 02000000
NAVAl: NIL VARIABLE
FAULT: LO
NEXT: 88
TOP: 89
BACKG: NIL GATE
NEXTG: 31
UNNO: NIL UNIT
PROB: 0.0200

EVENT 88
IDEN: T
UNID: 13 VALVE
VAR: X
INOI: 10000000
NAVAl: NIL VARIABLE
FAULT: LO
NEXT: 89
TOP: NIL EVENT
BACKG: 31
NEXTG: QM
UNNO: NIL UNIT
PROB: 0.0200

EVENT 89
IDEN: M
UNID: 13 VALVE
VAR: Q
INOI: 00000000
NAVAl: NIL VARIABLE
FAULT: NO-FLOW
NEXT: 93
TOP: 92
BACKG: NIL GATE
NEXTG: 32
UNNO: NIL UNIT
PROB: 0.0000
EVENT : 90
IDEN: T
UNID : 13 VALVE
VAR: Q
INOU : 1000000
NAVA: NIL VARIABLE
FAULT: NO-FLOW
NEXT: 91
TOP: NIL EVENT
BACKG: 32
NEXTG: QM
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 91
IDEN: B
UNID : 13 VALVE
VAR: S
INOU : 0200000
NAVA: NIL VARIABLE
FAULT: COMP-BLOC
NEXT: 92
TOP: NIL EVENT
BACKG: 32
NEXTG: NIL GATE
UNNO: NIL UNIT
PROB : 0.0001

EVENT : 92
IDEN: M
UNID : 13 VALVE
VAR: Q
INOU : 3000000
NAVA: NIL VARIABLE
FAULT: GT0
NEXT: 93
TOP: NIL EVENT
BACKG: NIL GATE
NEXTG: 33
UNNO: NIL UNIT
PROB : 0.0000

EVENT : 93
IDEN: T
UNID : 13 VALVE
VAR: Q
INOU : 1000000
NAVA: NIL VARIABLE
FAULT: GT0
NEXT: 94
TOP: NIL EVENT
BACKG: 33
NEXTG: QM
UNNO: NIL UNIT
PROB : 0.0000
ARRAY OF EVENTS

***************

EVENT : 94
IDEN: M
UNID : 9 PIPE
VAR: Q
INOU: 200000
NAVAl: 4
FAULT: LO
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: NIL GATE
NEXTG: 34
UNNO: 4
PROB : 0.0000

EVENT : 95
IDEN: T
UNID : 9 PIPE
VAR: P
INOU: 1000000
NAVAl: 3
FAULT: LO
NEXT: 110
TOP: NIL EVENT
BACKG: 34
NEXTG: 35
UNNO: 4
PROB : 3.0000

EVENT : 96
IDEN: T
UNID : 9 PIPE
VAR: Q
INOU: 1200000
NAVAl: 3
FAULT: LO
NEXT: 169
TOP: NIL EVENT
BACKG: 35
NEXTG: 36
UNNO: 4
PROB : 0.0000

EVENT : 97
IDEN: T
UNID : 13 VALVE
VAR: P
INOU: 1000000
NAVAl: 2
FAULT: LO
NEXT: 185
TOP: NIL EVENT
EVENT: 98
IDEN: T
UNID: 13
VALVE
VAR: Q
INOU: 18020000
NAVA: 2
FAULT: LO
NEXT: 104
TOP: NIL EVENT
BACKG: 37
NEXTG: 38
UNN0: 3
PROB: 0.0000

EVENT: 99
IDEN: T
UNID: 9
PIPE
VAR: P
INOU: 18020000
NAVA: 1
FAULT: LO
NEXT: 102
TOP: NIL EVENT
BACKG: 38
NEXTG: 39
UNN0: 2
PROB: 0.0000

EVENT: 100
IDEN: T
UNID: 9
PIPE
VAR: Q
INOU: 18020000
NAVA: 1
FAULT: LO
NEXT: 101
TOP: NIL EVENT
BACKG: 39
NEXTG: NIL GATE
UNN0: 2
PROB: 0.0000

EVENT: 101
IDEN: B
UNID: 9
PIPE
VAR: S
INOU: 82000000
NAVA: NIL VARIABLE
FAULT: LK-LP-ENV
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: 39
EVENT: 102
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 02000000
NAV: NIL VARIABLE
FAULT: BLOCKAGE
NEXT: 103
TOP: NIL EVENT
BACKG: 38
NEXTG: NIL GATE
UNN: 2
PROB: 8.0005

EVENT: 103
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 02000000
NAV: NIL VARIABLE
FAULT: LK-LP-ENV
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: 38
NEXTG: NIL GATE
UNN: 2
PROB: 8.0022

EVENT: 104
IDEN: B
UNID: 13 VALVE
VAR: S
INOU: 02000000
NAV: NIL VARIABLE
FAULT: LK-LP-ENV
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: 37
NEXTG: NIL GATE
UNN: 3
PROB: 8.0007

EVENT: 105
IDEN: R
UNID: 13 VALVE
VAR: S
INOU: 02000000
NAV: NIL VARIABLE
FAULT: CLOSED
NEXT: 106
TOP: NIL EVENT
BACKG: 36
NEXTG: 40
EVENT : 106
IDEN: B
UNID: 13 VALUE
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: BLOCKAGE
NEXT: 107
TOP: NIL EVENT
BACKG: 40
NEXTG: NIL GATE
UNNO: 3
PROB: 0.0000

EVENT : 107
IDEN: B
UNID: 13 VALUE
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: SHUT
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: 40
NEXTG: NIL GATE
UNNO: 3
PROB: 0.0005

EVENT : 108
IDEN: B
UNID: 13 VALUE
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: LK-LP-ENV
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: 36
NEXTG: NIL GATE
UNNO: 3
PROB: 0.0003

EVENT : 109
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: LK-LP-ENV
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: 35
NEXTG: NIL GATE
UNNO: 4
EVENT : 110
IDEN: T
UNID: 9 PIPE
VAR: P
INOU: 00000000
NAVA: 4
FAULT: HI
NEXT: 111
TOP: NIL EVENT
BACKG: 34
NEXTG: NIL GATE
UNNO: 4
PROB: 0.0035

EVENT : 111
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 22000000
NAVA: NIL VARIABLE
FAULT: BLOCKAGE
NEXT: 112
TOP: NIL EVENT
BACKG: 34
NEXTG: NIL GATE
UNNO: 4
PROB: 0.0002

EVENT : 112
IDEN: B
UNID: 9 PIPE
VAR: S
INOU: 02000000
NAVA: NIL VARIABLE
FAULT: LK-LF-ENV
NEXT: NIL EVENT
TOP: NIL EVENT
BACKG: 34
NEXTG: NIL GATE
UNNO: 4
PROB: 0.0025
ARRAY OF FAULTS

FAULT (1) : HI
FAULT (2) : LO
FAULT (3) : OPEN
FAULT (4) : CLOSED
FAULT (5) : FL-EX-ENV
FAULT (6) : BLOCKAGE
FAULT (7) : LK-HP-ENV
FAULT (8) : LK-HP-ENV
FAULT (9) : WIDE-OPEN
FAULT (10) : SHUT
FAULT (11) : BLOC-OUTL
FAULT (12) : BLOC-INLE
FAULT (13) : FAI-TO-OP
FAULT (14) : MANUAL
FAULT (15) : CONT-STCK
FAULT (16) : SENS-STCK
FAULT (17) : VALV-STCK
FAULT (18) : FAI-TO-CL
FAULT (19) : SET-PO-HI
FAULT (20) : FAIL-OPEN
FAULT (21) : FAIL-HI
FAULT (22) : SEN-FA-LO
FAULT (23) : SET-PO-LO
FAULT (24) : FAI-CLOSE
FAULT (25) : FAIL-LO
FAULT (26) : SEN-FA-HI
FAULT (27) : CONT-F-HI
FAULT (28) : CONT-F-LO
FAULT (29) : NO-CHANGE
FAULT (30) : EXT-FIRE
FAULT (31) : MECH-FAIL
FAULT (32) : OTHER-CAU
FAULT (33) : Z-HI
FAULT (34) : Z-LO
FAULT (35) : A
FAULT (36) : B
FAULT (37) : C
FAULT (38) : D
FAULT (39) : DUMMY
FAULT (40) : NO-FLOW
FAULT (41) : SI-STR-PL
FAULT (42) : NO-SIGNAL
FAULT (43) : GTB
FAULT (44) : SHUTDOWN
FAULT (45) : COMP-BLOC
FAULT (46) :
FAULT (47) :
FAULT (48) :
FAULT (49) :
FAULT (50) :
ARRAY OF GATES

**************

GATE : 1
CLASS: 0 0
EXTRA: 5
BACK: 1
NEXT: 2
NEGATE: 2
BC : 1

GATE : 2
CLASS: 0 0
EXTRA: 5
BACK: 3
NEXT: 4
NEGATE: 3
BC : 2

GATE : 3
CLASS: 0 0
EXTRA: 5
BACK: 5
NEXT: 6
NEGATE: 4
BC : 3

GATE : 4
CLASS: 0 0
EXTRA: 5
BACK: 7
NEXT: 8
NEGATE: 5
BC : 4

GATE : 5
CLASS: 0 0
EXTRA: 5
BACK: 9
NEXT: 10
NEGATE: 6
BC : 5

GATE : 6
CLASS: 0 0
EXTRA: 5
BACK: 11
NEXT: 12
NEGATE: 7
BC : 6
GATE : 14
CLASS: 0  0
EXTRA: 5
BACKE: 29
NEXTE: 30
NEGATE: 15
BC : 14

GATE : 15
CLASS: 0  0
EXTRA: 5
BACKE: 34
NEXTE: 35
NEGATE: 16
BC : 15

GATE : 16
CLASS: 0  0
EXTRA: 5
BACKE: 39
NEXTE: 40
NEGATE: 17
BC : 16

GATE : 17
CLASS: 0  0
EXTRA: 5
BACKE: 43
NEXTE: 44
NEGATE: 18
BC : 17

GATE : 18
CLASS: 0  0
EXTRA: 5
BACKE: 46
NEXTE: 47
NEGATE: 19
BC : 18

GATE : 19
CLASS: 0  0
EXTRA: 5
BACKE: 48
NEXTE: 49
NEGATE: 20
BC : 19

GATE : 20
CLASS: 0  0
EXTRA: 5
BACKE: 50
NEXTE: 51
NEGATE: 21
BC : 20

GATE : 21
CLASS: 0 0
EXTRA: 5
BACKE: 52
NEXTE: 53
NEGATE: 22
BC : 21

GATE : 22
CLASS: 0 0
EXTRA: 5
BACKE: 55
NEXTE: 56
NEGATE: 23
BC : 22

GATE : 23
CLASS: 0 0
EXTRA: 5
BACKE: 57
NEXTE: 58
NEGATE: 24
BC : 23

GATE : 24
CLASS: 0 0
EXTRA: 5
BACKE: 62
NEXTE: 63
NEGATE: 25
BC : 24

GATE : 25
CLASS: 0 0
EXTRA: 5
BACKE: 67
NEXTE: 68
NEGATE: 26
BC : 25

GATE : 26
CLASS: 0 0
EXTRA: 5
BACKE: 70
NEXTE: 71
NEGATE: 27
BC : 26

GATE : 27
CLASS: 0 0
EXTRA: 5
ARRAY OF GATES

***************

GATE : 34
CLASS: 0 0
EXTRA: S
BACKE: 94
NEXTE: 95
NEGATE: 35
BC : 34

GATE : 35
CLASS: 0 0
EXTRA: S
BACKE: 95
NEXTE: 96
NEGATE: 36
BC : 35

GATE : 36
CLASS: 0 0
EXTRA: S
BACKE: 96
NEXTE: 97
NEGATE: 37
BC : 36

GATE : 37
CLASS: 0 0
EXTRA: S
BACKE: 97
NEXTE: 98
NEGATE: 38
BC : 37

GATE : 38
CLASS: 0 0
EXTRA: S
BACKE: 98
NEXTE: 99
NEGATE: 39
BC : 38

GATE : 39
CLASS: 0 0
EXTRA: S
BACKE: 99
NEXTE: 100
NEGATE: 40
BC : 39
GATE: 40
CLASS: 0 0
EXTRA: S
BACK: 105
NEXT: 106
NEGATE: 41
EC: 40
ARRAY OF DATA FOR VARIABLES

*************

DAT : 1
HL : 132
LL : 0
PRVA : 0
ACTVA : 0
Prio : 1
PREX : 5

DAT : 2
HL : 100
LL : 0
PRVA : 0
ACTVA : 0
Prio : 2
PREX : 5

DAT : 3
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 4
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 5
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 6
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5
<table>
<thead>
<tr>
<th>DAT</th>
<th>HL</th>
<th>LL</th>
<th>PRVA</th>
<th>ACTVA</th>
<th>PRIO</th>
<th>PREX</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>DAT</td>
<td>HL</td>
<td>LL</td>
<td>PRVA</td>
<td>ACTVA</td>
<td>P Rio</td>
<td>PREX</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prio : 32
PREX : 5

DAT : 21
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 22
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 23
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 24
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 25
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 26
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
Prio : 32
PREX : 5

DAT : 27
HL : 0
LL : 0

PAGE: MECURV: PRO
PRVA : 0
ACTVA : 0
PRIO : 32
PREX : 5

DAT : 28
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
PRIO : 32
PREX : 5

DAT : 29
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
PRIO : 32
PREX : 5

DAT : 30
HL : 0
LL : 0
PRVA : 0
ACTVA : 0
PRIO : 32
PREX : 5
ARRAY OF STREAMS

STREAM: 1
SI: 0
EI: 0 1
FROM: 1
TO: 2
NAV: 1
NEXTS: NIL STREAM

STREAM: 2
SI: 1
EI: 0 0
FROM: 1
TO: 2
NAV: NIL VARIABLE
NEXTS: NIL STREAM

STREAM: 3
SI: 0
EI: 0 1
FROM: 2
TO: 3
NAV: 3
NEXTS: NIL STREAM

STREAM: 4
SI: 1
EI: 0 0
FROM: 2
TO: 3
NAV: NIL VARIABLE
NEXTS: NIL STREAM

STREAM: 5
SI: 0
EI: 0 1
FROM: 3
TO: 4
NAV: 5
NEXTS: NIL STREAM

STREAM: 6
SI: 1
EI: 0 0
FROM: 3
TO: 4
NAV: NIL VARIABLE
NEXTS: NIL STREAM
STREAM: 7
SI: 0
EO: 0 1
FROMU: 4
TOU: 5
NAV: 7
NEXTS: NIL STREAM

STREAM: 8
SI: 1
EO: 0 0
FROMU: 4
TOU: 5
NAV: NIL VARIABLE
NEXTS: NIL STREAM

STREAM: 9
SI: 2
EO: 2
FROMU: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 10

STREAM: 10
SI: 2
EO: 2
FROMU: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 11

STREAM: 11
SI: 2
EO: 2
FROMU: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 12

STREAM: 12
SI: 2
EO: 2
FROMU: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 13

STREAM: 13
SI: 2
EO: 2
FROMU: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 14
STREAM: 14
SI0: 2
EO: 2
FROM: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 15

STREAM: 15
SI0: 2
EO: 2
FROM: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 16

STREAM: 16
SI0: 2
EO: 2
FROM: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 17

STREAM: 17
SI0: 2
EO: 2
FROM: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 18

STREAM: 18
SI0: 2
EO: 2
FROM: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 19

STREAM: 19
SI0: 2
EO: 2
FROM: NIL UNIT
TOU: NIL UNIT
NAV: NIL VARIABLE
NEXTS: 20

STREAM: 20
SI0: 2
EO: 2
FROM: NIL UNIT
TOU: NIL UNIT
NAVAR: NIL VARIABLE
NESTS: 21

STREAM: 21
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL Unit
NAVAR: NIL VARIABLE
NESTS: 22

STREAM: 22
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NESTS: 23

STREAM: 23
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NESTS: 24

STREAM: 24
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NESTS: 25

STREAM: 25
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NESTS: 26

STREAM: 26
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL Unit
NAVAR: NIL VARIABLE
NESTS: 27

STREAM: 27
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 28

STREAM: 28
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 29

STREAM: 29
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 30

STREAM: 30
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 31

STREAM: 31
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 32

STREAM: 32
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 33

STREAM: 33
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 34

STREAM: 34
STREAM: 35
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 35

STREAM: 36
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 36

STREAM: 37
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 37

STREAM: 38
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 38

STREAM: 39
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 39

STREAM: 40
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 40

STREAM: 41
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 41
STREAM: 41
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVARI: NIL VARIABLE
NEXTS: 42

STREAM: 42
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVARI: NIL VARIABLE
NEXTS: 43

STREAM: 43
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVARI: NIL VARIABLE
NEXTS: 44

STREAM: 44
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVARI: NIL VARIABLE
NEXTS: 45

STREAM: 45
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVARI: NIL VARIABLE
NEXTS: 46

STREAM: 46
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVARI: NIL VARIABLE
NEXTS: 47

STREAM: 47
SI0: Z
EIO: Z
FROMU: NIL UNIT
TOUN: NIL UNIT
NAVARI: NIL VARIABLE
STREAM: 48
SIZE: Z
EIO: Z
FROMU: NIL UNIT
TOWN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 49

STREAM: 49
SIZE: Z
EIO: Z
FROMU: NIL UNIT
TOWN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: 50

STREAM: 50
SIZE: Z
EIO: Z
FROMU: NIL UNIT
TOWN: NIL UNIT
NAVAR: NIL VARIABLE
NEXTS: NIL NEXT STREAM
ARRAY TO LOCATE TOP EVENTS

*************************

LOT: 1
TYPE: 'DUMMY-H
TEX: 5
FIRSEL: 1

LOT: 2
TYPE: 'DUMMY-T
TEX: 5
FIRSEL: 21

LOT: 3
TYPE: PIPE
TEX: 5
FIRSEL: 25

LOT: 4
TYPE: VALVE
TEX: 5
FIRSEL: 57

LOT: 5
TYPE: 
TEX: 5
FIRSEL: NIL EVENT

LOT: 6
TYPE: 
TEX: 5
FIRSEL: NIL EVENT

LOT: 7
TYPE: 
TEX: 5
FIRSEL: NIL EVENT

LOT: 8
TYPE: 
TEX: 5
FIRSEL: NIL EVENT

LOT: 9
TYPE: 
TEX: 5
FIRSEL: NIL EVENT

---

TOPEV.PRO
LOT: 13
TYPE: S
TEX: S
FIRSEL: NIL EVENT

LOT: 11
TYPE: S
TEX: S
FIRSEL: NIL EVENT

LOT: 12
TYPE: S
TEX: S
FIRSEL: NIL EVENT

LOT: 13
TYPE: S
TEX: S
FIRSEL: NIL EVENT

LOT: 14
TYPE: S
TEX: S
FIRSEL: NIL EVENT

LOT: 15
TYPE: S
TEX: S
FIRSEL: NIL EVENT
### Array of Type of Units

<table>
<thead>
<tr>
<th>Type of Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Central Pump</td>
</tr>
<tr>
<td>(2)</td>
<td>Closed Tank</td>
</tr>
<tr>
<td>(3)</td>
<td>Control Valve</td>
</tr>
<tr>
<td>(4)</td>
<td>Control Valve</td>
</tr>
<tr>
<td>(5)</td>
<td>Controller</td>
</tr>
<tr>
<td>(6)</td>
<td>Dummy Unit</td>
</tr>
<tr>
<td>(7)</td>
<td>Dummy Unit</td>
</tr>
<tr>
<td>(8)</td>
<td>Heat Exchanger</td>
</tr>
<tr>
<td>(9)</td>
<td>Pipe</td>
</tr>
<tr>
<td>(10)</td>
<td>Sensor-Q</td>
</tr>
<tr>
<td>(11)</td>
<td>Sensor-P</td>
</tr>
<tr>
<td>(12)</td>
<td>Sensor-T</td>
</tr>
<tr>
<td>(13)</td>
<td>Valve</td>
</tr>
<tr>
<td>(14)</td>
<td></td>
</tr>
<tr>
<td>(15)</td>
<td></td>
</tr>
</tbody>
</table>
ARRAY OF UNITS

UNIT: 1
UNUM: 1
TY: DUMMY-H
TVE: S
INST: NIL
OUTST: 1
NEXU: NIL UNIT

UNIT: 2
UNUM: 2
TY: PIPE
TVE: S
INST: 2
OUTST: 3
NEXU: NIL UNIT

UNIT: 3
UNUM: 3
TY: VALVE
TVE: S
INST: 4
OUTST: 5
NEXU: NIL UNIT

UNIT: 4
UNUM: 4
TY: PIPE
TVE: S
INST: 6
OUTST: 7
NEXU: NIL UNIT

UNIT: 5
UNUM: 5
TY: DUMMY-T
TVE: S
INST: 8
OUTST: NIL
NEXU: NIL UNIT

UNIT: 6
UNUM: 6
TY: S
TVE: S
INST: NIL STREAM
OUTST: NIL STREAM
NEXU: 7
UNIT: 14
NUM: 3
TY: S
TYE: S
INST: NIL STREAM
OUTST: NIL STREAM
NEXU: 15

UNIT: 15
NUM: 3
TY: S
TYE: S
INST: NIL STREAM
OUTST: NIL STREAM
NEXU: NIL NEXT UNIT
ARRAY OF VARIABLES

VAR: 1
NAMEF: P
NAMET: P
MESU : N
EX : 3
NOVA: 1
BACKS: 1
NEXVAR: 2
VALUES : NILDAT

VAR: 2
NAMEF: Q
NAMET: Q
MESU : N
EX : 3
NOVA: 1
BACKS: 1
NEXVAR: NIL VARIABLE
VALUES : NILDAT

VAR: 3
NAMEF: P
NAMET: P
MESU : N
EX : 3
NOVA: 2
BACKS: 3
NEXVAR: 4
VALUES : NILDAT

VAR: 4
NAMEF: Q
NAMET: Q
MESU : N
EX : 3
NOVA: 2
BACKS: 3
NEXVAR: NIL VARIABLE
VALUES : NILDAT

VAR: 5
NAMEF: P
NAMET: P
MESU : Y
EX : 3
NOVA: 3
BACKS: 5
NEXVAR: 6
VALUES : 1
<table>
<thead>
<tr>
<th>VAR</th>
<th>NAMEF</th>
<th>NAMET</th>
<th>MESU</th>
<th>EX</th>
<th>NOVA</th>
<th>BACKS</th>
<th>NEXVAR</th>
<th>VALUES</th>
<th>NILDAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Q</td>
<td>Q</td>
<td>N</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>NIL VARIABLE</td>
<td>2</td>
<td>NILDAT</td>
</tr>
<tr>
<td>7</td>
<td>P</td>
<td>P</td>
<td>N</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>NIL VARIABLE</td>
<td>2</td>
<td>NILDAT</td>
</tr>
<tr>
<td>8</td>
<td>Q</td>
<td>Q</td>
<td>Y</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>NIL VARIABLE</td>
<td>2</td>
<td>NILDAT</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>2</td>
<td>0</td>
<td>NIL STREAM</td>
<td>10</td>
<td>NILDAT</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>2</td>
<td>0</td>
<td>NIL STREAM</td>
<td>10</td>
<td>NILDAT</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>2</td>
<td>0</td>
<td>NIL STREAM</td>
<td>10</td>
<td>NILDAT</td>
<td></td>
</tr>
</tbody>
</table>
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 12
VALUES: 'NILDAT

VAR: 12
NAMEF: S
NAMEF: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 13
VALUES: 'NILDAT

VAR: 13
NAMEF: S
NAMEF: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 14
VALUES: 'NILDAT

VAR: 14
NAMEF: S
NAMEF: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 15
VALUES: 'NILDAT

VAR: 15
NAMEF: S
NAMEF: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 16
VALUES: 'NILDAT

VAR: 16
NAMEF: S
NAMEF: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 17
VALUES: 'NILDAT
VAR: 17
NAMEF: S
NAMET: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 18
VALUES: "NILDAT"

VAR: 18
NAMEF: S
NAMET: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 19
VALUES: "NILDAT"

VAR: 19
NAMEF: S
NAMET: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 20
VALUES: "NILDAT"

VAR: 20
NAMEF: S
NAMET: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 21
VALUES: "NILDAT"

VAR: 21
NAMEF: S
NAMET: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 22
VALUES: "NILDAT"

VAR: 22
NAMEF: S
NAMET: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 23
VALUES : NILDAT

VAR: 23
NAMEF: S
NAMET: S
MESU : S
EX : Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 24
VALUES : NILDAT

VAR: 24
NAMEF: S
NAMET: S
MESU : S
EX : Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 25
VALUES : NILDAT

VAR: 25
NAMEF: S
NAMET: S
MESU : S
EX : Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 26
VALUES : NILDAT

VAR: 26
NAMEF: S
NAMET: S
MESU : S
EX : Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 27
VALUES : NILDAT

VAR: 27
NAMEF: S
NAMET: S
MESU : S
EX : Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 28
VALUES : NILDAT

VAR: 28

PAGE

VARIAB-PRO
NAMEF: S
NAMEF: S
NAMEF: S
EX: 12
NOVA: Ø
BACKS: NIL STREAM
NEXVAR: 29
VALUES: NILDAT

VAR: 29
NAMEF: S
NAMEF: S
NAMEF: S
EX: 12
NOVA: Ø
BACKS: NIL STREAM
NEXVAR: 30
VALUES: NILDAT

VAR: 30
NAMEF: S
NAMEF: S
NAMEF: S
EX: 12
NOVA: Ø
BACKS: NIL STREAM
NEXVAR: 31
VALUES: NILDAT

VAR: 31
NAMEF: S
NAMEF: S
NAMEF: S
EX: 12
NOVA: Ø
BACKS: NIL STREAM
NEXVAR: 32
VALUES: NILDAT

VAR: 32
NAMEF: S
NAMEF: S
NAMEF: S
EX: 12
NOVA: Ø
BACKS: NIL STREAM
NEXVAR: 33
VALUES: NILDAT

VAR: 33
NAMEF: S
NAMEF: S
NAMEF: S
EX: 12
NOVA: Ø
BACKS: NIL STREAM
NAMEF: S
MESU: S
EX: 1
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 43
VALUES: NILDAT

VAR: 43
NAMEF: S
NAMEF: S
MESU: S
EX: 2
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 44
VALUES: NILDAT
VALUES : NILDAT

VAR: 45
NAMEF: S
NAMET: S
MESU: S
EX: Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 46
VALUES : NILDAT

VAR: 46
NAMEF: S
NAMET: S
MESU: S
EX: Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 47
VALUES : NILDAT

VAR: 47
NAMEF: S
NAMET: S
MESU: S
EX: Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 48
VALUES : NILDAT

VAR: 48
NAMEF: S
NAMET: S
MESU: S
EX: Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 49
VALUES : NILDAT

VAR: 49
NAMEF: S
NAMET: S
MESU: S
EX: Z
NOVA: 0
BACKS: NIL STREAM
NEXVAR: 50
VALUES : NILDAT

VAR: 50
NAMEF: S
NAMET: S

PAGE

VARI AB.PRO
MESU: S
EX: Z
NOVA: Ø
BACKS: NIL STREAM
NEXTVAR: NIL NEXT VAR
VALUES: NILDAT